Capitolo 1

Introduzione

Benvenuti

Vi auguriamo una piacevole lettura del manuale PHP, il libro che insegnerà a creare siti Web dinamici, che interagiscano con gli utenti, e renderà possibile lo sviluppo di applicazioni Web complesse.

Informazioni su PHP

PHP (*PHP Hypertext Pre-processor*) è un linguaggio di scripting che viene incorporato nell'HTML (*HyperText Markup Language*). L'obiettivo del linguaggio è consentire la costruzione rapida e semplice di pagine Web dinamiche. PHP funziona insieme a un server Web e può essere utilizzato con moltissimi sistemi operativi, tra cui Microsoft Windows e UNIX.

PHP è diverso dagli altri script CGI (*Common Gateway Interface*), scritti in linguaggi come Perl o C, poiché questi richiedono all'utente di creare programmi separati che producono codice HTML. PHP invece è incorporato nel documento HTML con speciali tag iniziali e finali che permettono di entrare e uscire da esso. In questo modo, l'utente finale può visualizzare rapidamente le pagine e contare su un'elevata protezione e trasparenza. Con PHP potete fare tutto quello che otterreste scrivendo applicazioni CGI separate, vale a dire creare pagine Web dinamiche, elaborare moduli e gestire file.

La sintassi di PHP è simile a quella dei linguaggi di programmazione C, C++ e Java. Per chi ha una certa dimestichezza con questi linguaggi, PHP sembrerà un linguaggio molto familiare. Chi invece non ha alcuna esperienza di tali linguaggi non deve preoccuparsi, poiché PHP è molto semplice da comprendere.

Una tra le caratteristiche più importanti ed efficaci di PHP è la sua capacità di interfacciarsi con una vasta gamma di database. Al momento sono supportati più di venti database diversi, consentendo agli sviluppatori PHP di creare agevolmente pagine Web in grado di interagire con database. Più avanti vedrete come utilizzare il database MySQL.

Questo libro fa al caso vostro?

Nella realizzazione di questo manuale si è cercato di concentrarsi su un tipo di utente target. Se siete:

1. ingegneri del software professionisti, desiderosi di applicare le vostre capacità di sviluppatori alla creazione di applicazioni Web;
2. sviluppatori professionisti di siti Web, desiderosi di imparare a creare applicazioni Web dinamiche più potenti con PHP;
3. sviluppatori intranet che lavorano per una società che vuole espandere le proprie applicazioni di rete servendosi di una tecnologia Web dinamica;
4. neolaureti che devono imparare a programmare con PHP in breve tempo;
5. studenti iscritti a un corso pre-laurea o post-laurea cui viene chiesto di sviluppare un sistema Web dinamico come parte integrante di un progetto;

questo libro fa al caso vostro.

Requisiti indispensabili...

Partiamo dal presupposto che:

1. sappiate utilizzare molto bene un computer Microsoft Windows o UNIX;
2. sappiate realizzare pagine Web con HTML;
3. abbiate una certa dimestichezza con linguaggi di scripting come Javascript;
4. magari una discreta esperienza di programmazione con linguaggi quali C, Visual Basic, Java o COBOL.

Se possedete i requisiti appena elencati, questo manuale è stato scritto per voi.

Utilizzo del manuale

Questo manuale è stato scritto per essere letto dall'inizio alla fine. Gli ultimi capitoli si basano sulle conoscenze e gli spunti introdotti nei primi. Dopo aver letto il libro, potrete fare riferimento ai singoli capitoli qualora voleste rinfrescarvi la memoria su un particolare aspetto di PHP.

Struttura del manuale

Il libro è suddiviso in capitoli che trattano aspetti specifici del linguaggio PHP. I capitoli sono raggruppati a loro volta in parti che fanno riferimento ad ambiti più generici del linguaggio.

La Parte 1 introduce il linguaggio PHP, spiega in che modo configurare un ambiente di sviluppo e presenta i primi script PHP.

- Il Capitolo 2 fornisce un'introduzione e una descrizione sommaria di PHP.
- Il Capitolo 3 spiega come creare un ambiente di sviluppo PHP e mostra dove trovare aiuto e assistenza.
- Il Capitolo 4 introduce il primo programma PHP.
- Il Capitolo 5 si occupa dei principi basilari di PHP.

La Parte 2 introduce le istruzioni, le variabili e le espressioni di base del linguaggio PHP.

- Il Capitolo 6 presenta le variabili e i tipi di PHP.
- Il Capitolo 7 analizza i tipi booleani, interi e in virgola mobile.
- Il Capitolo 8 presenta il tipo stringa.
- Il Capitolo 9 chiarisce il concetto di variabili PHP predefinite e spiega come valutarle.
- Il Capitolo 10 tratta le espressioni, gli operandi e gli operatori.
- Il Capitolo 11 illustra le istruzioni `if` e `switch` e mostra come si possano utilizzare gli operatori condizionali e booleani all'interno di tali istruzioni.
- Il Capitolo 12 descrive i vari cicli utilizzabili con PHP.
- Il Capitolo 13 analizza i concetti di scomposizione funzionale, creazione di funzioni e file inclusi.
- Il Capitolo 14 illustra il concetto di array.
- Il Capitolo 15 introduce le date, le ore e i numeri casuali e spiega come accedervi.

La Parte 3 esamina come manipolare le stringhe e gli array con numerose funzioni PHP standard.

- Il Capitolo 16 si occupa della manipolazione degli array.
- Il Capitolo 17 analizza la manipolazione delle stringhe.

La Parte 4 introduce il concetto di interazione con l'utente e spiega come gli utenti possano interagire con le applicazioni PHP tramite i moduli.

- Il Capitolo 18 spiega come gli script possano interagire con l'utente.
- Il Capitolo 19 descrive la convalida dei dati dei moduli e la conservazione dei dati.
- Il Capitolo 20 tratta i problemi legati alla protezione dei dati dei moduli.
- Il Capitolo 21 spiega come usare PHP per inviare messaggi di posta elettronica agli utenti.

La Parte 5 descrive due metodi che permettono di conservare i dati nel passaggio da una pagina Web all'altra, vale a dire cookie e sessioni.

- Il Capitolo 22 presenta i cookie.
- Il Capitolo 23 descrive le sessioni e la gestione delle sessioni.

La Parte 6 analizza in che modo gli script possano gestire gli errori e mostra come bufferizzare l'output sulle pagine Web e regolare il contenuto del buffer.

- Il Capitolo 24 analizza la gestione degli errori.
- Il Capitolo 25 spiega come bufferizzare e modificare l'output delle pagine Web.

La Parte 7 espone il concetto di gestione dei file.

- Il Capitolo 26 descrive come leggere e scrivere dati su file.
- Il Capitolo 27 analizza in dettaglio altre operazioni che è possibile eseguire con i file.
- Il Capitolo 28 spiega come caricare i file su un server tramite i moduli.

La Parte 8 introduce il concetto di manipolazione grafica.

- Il Capitolo 29 analizza esempi di manipolazione grafica semplice.
- Il Capitolo 30 introduce la libreria GD e illustra come utilizzarla per creare in modo dinamico nuove immagini.
- Il Capitolo 31 illustra come utilizzare la libreria GD per creare un'utile applicazione per la generazione di grafici.

La Parte 9 analizza il concetto di creazione di documenti diversi dalle pagine Web attraverso l'estensione della libreria PDF.

- Il Capitolo 32 si occupa della creazione di documenti PDF.

La Parte 10 esamina il concetto di collegamento di un database a un sito Web con il sistema di gestione database MySQL.

- Il Capitolo 33 offre un'introduzione alla progettazione di tabelle di database.
- Il Capitolo 34 spiega come installare il database MySQL e PHPmyadmin.
- Il Capitolo 35 spiega come configurare il database MySQL.
- Il Capitolo 36 descrive come utilizzare PHP per interfacciarlo a un database MySQL.

La Parte 11 descrive il paradigma a oggetti e spiega come usare PHP per creare applicazioni a oggetti.

- Il Capitolo 37 presenta i concetti di classi e oggetti, caratteristici del paradigma a oggetti.
- Il Capitolo 38 prosegue con l'analisi dell'orientamento agli oggetti aggiungendo un'analisi dell'ereditarietà delle classi.

Infine, la Parte 12 descrive un'applicazione di e-commerce realizzata mettendo in pratica tutti i concetti presentati nei capitoli precedenti del manuale.

- Il Capitolo 39 descrive il sistema di e-commerce front-end.
- Il Capitolo 40 descrive il sistema di amministrazione del sito di e-commerce.

Script PHP

Questo libro contiene numerosi script d'esempio; tutti gli script sono stati verificati con l'ultima versione dell'ambiente PHP. Tali script possono essere prelevati dal sito Web dedicato al manuale, `www.phpmysql-manual.com`, risparmiandovi la fatica di inserirli. Si deve ammettere, però, che molto spesso si impara di più digitando manualmente gli script degli esempi, quindi potete tranquillamente scegliere di adottare questa tecnica.

Partenza...

Non rimane molto altro da dire se non invitarvi a girare la pagina e procedere con il prossimo capitolo.

Capitolo 2

Introduzione a PHP

Introduzione

In questo capitolo presenteremo PHP, descrivendo che cos'è, la sua storia e la sua popolarità. Iniziamo analizzando la vera natura di PHP.

Che cos'è PHP?

"PHP (acronimo ricorsivo di *PHP Hypertext Preprocessor*) è un linguaggio di scripting open-source per la realizzazione di pagine Web dinamiche, applicazioni di e-commerce e altre applicazioni Web" (definizione tradotta da `www.zend.com/zend/aboutphp.php`). Le pagine Web dinamiche sono quelle che non rimangono inalterate, ma interagiscono con l'utente permettendogli di vivere un'esperienza Web più ricca e interessante. I sistemi Web dinamici sono utilizzati soprattutto nei sistemi di commercio elettronico che interagiscono con i database, dando agli utenti la possibilità di selezionare ed acquistare online vari prodotti.

PHP è disponibile gratuitamente al sito `www.php.net` e costituisce una soluzione semplice ma potente per sviluppare sistemi Web dinamici. PHP è incorporato nelle pagine HTML, e ciò consente di inserire le istruzioni di script esattamente nel punto in cui sono necessarie.

PHP è ampiamente supportato da una vasta comunità online ed è un prodotto open-source. Tale comunità offre un supporto straordinario agli sviluppatori e qualsiasi errore viene individuato e risolto in breve tempo, poiché il nucleo fondamentale del codice PHP viene costantemente migliorato e aggiornato.

PHP presenta un'eccellente connettività con un gran numero di database (per esempio Oracle e MySQL), nonché l'integrazione con numerose librerie esterne che permettono al linguaggio di generare, per esempio, immagini grafiche e documenti PDF.

Infine PHP è indipendente dalla piattaforma, in quanto viene eseguito senza alcun problema su piattaforme sia Windows sia UNIX.

Storia di PHP

La storia di PHP risale al 1995. Da allora, modifiche e migliorie significative hanno portato al rilascio della versione 5.0 di PHP nel giugno 2003. La Tabella 2.1 è un adattamento delle informazioni disponibili all'indirizzo `www.php.net/manual/en/history.php` e descrive le release più importanti di questo linguaggio dall'anno del suo concepimento.

Tabella 2.1 Storia di PHP

Versione	Descrizione
PHP/FI Data: 1995	Nel 1995 Rasmus Lerdorf creò un insieme di script Perl per tenere traccia degli accessi al suo sistema Web. All'inizio, questi script erano noti con il nome di *"Personal Home Page Tools"*. Gli script originali sono stati riscritti con il linguaggio di programmazione C e sono stati migliorati con l'aggiunta di funzionalità superiori. Per esempio, la comunicazione con il database e la possibilità di sviluppare semplici applicazioni Web. Questa versione è nota con il nome di *Personal Home Page/Forms Interpreter* ed è stata messa a disposizione del pubblico.
PHP/FI 2.0 Data: 1997	Nel novembre 1997 venne rilasciato ufficialmente PHP/FI 2.0. Allora, 50.000 domini annunciarono di aver installato questo software.
PHP 3.0 Data: 1997	PHP 3.0 è la prima versione di PHP che può essere considerata simile a quella odierna. Venne creata da Andi Gutmans e Zeev Suraki come riscrittura di PHP/FI 2.0 per un progetto di e-commerce di un'università. Andi Gutmans e Zeev Suraki decisero di collaborare con Rasmus Lerdorf e annunciarono PHP 3.0 come successore ufficiale di PHP/FI 2.0. Le caratteristiche chiave di PHP 3.0 erano la straordinaria estensibilità, la connettività a database e il supporto per il paradigma a oggetti. Il linguaggio PHP 3.0 venne rilasciato con il nuovo nome PHP, che è l'acronimo ricorsivo di *PHP Hypertext Preprocessor*. Entro la fine del 1998 PHP 3.0 era installato su circa il 10% dei server Web presenti su Internet.
PHP 4 Data: 1999	Nell'inverno del 1998, Andi Gutmans e Zeev Suraki iniziarono a riscrivere il nucleo fondamentale di PHP per migliorare le prestazioni delle applicazioni più complesse. Il nuovo motore principale prese il nome di *"Zend Engine"* come derivazione dai nomi dei due autori "Zeev e Andi".
PHP 5 Data: 2003	Rilasciato in versione beta a metà del 2003, PHP 5.0 è la release più recente del linguaggio, e ha come nucleo fondamentale il nuovo Zend Engine 2.0.

Quanto è diffuso PHP?

La diffusione di PHP è grande e in costante crescita. Spesso è difficile stabilire con precisione il grado di notorietà di un linguaggio di programmazione, poiché esistono molti modi diversi per effettuare una stima. Tuttavia, il sondaggio di Netcraft fornisce un calcolo del numero di indirizzi di provider Internet e di domini che utilizzano PHP. La Figura 2.1 mostra le ultime statistiche del sondaggio Netcraft (`www.php.net/usage.php`), che illustrano chiaramente la crescita dell'uso di PHP dal gennaio 1999.

Un'altra misurazione, il TPCI (*TIOBE Programming Community Index*), dà un'indicazione della popolarità dei linguaggi di programmazione. A luglio del 2003 PHP aveva raggiunto la sesta posizione tra i linguaggi più noti attualmente in uso, come mostrato nel grafico della Figura 2.2.

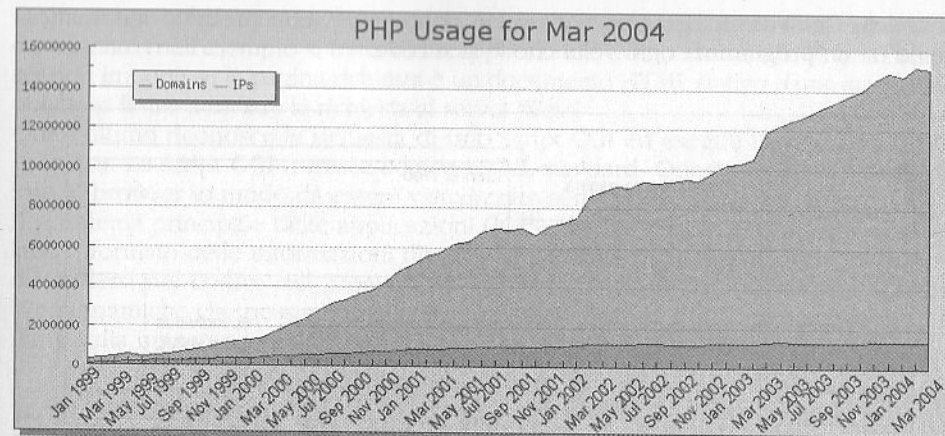


Figura 2.1
Il sondaggio di Netcraft.

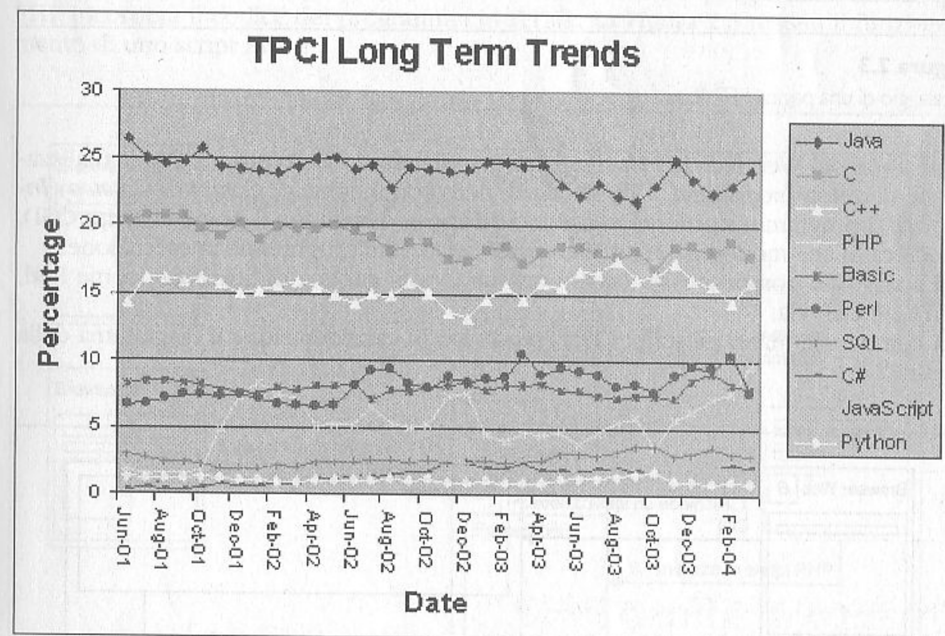


Figura 2.2
TPCI. Popolarità dei linguaggi di programmazione.

Server Web e CGI

Uno dei problemi maggiori con i documenti HTML è legato alla loro staticità: infatti tali documenti visualizzano sempre il medesimo contenuto ogni volta che vi si accede. La Figura 2.3 mostra un server Web tradizionale che riceve una richiesta relativa a una pagina Web specifica e passa tale richiesta al browser Web. Le pagine Web

dinamiche sono invece pagine Web il cui contenuto viene prodotto automaticamente da un programma ogni volta che vi si accede.

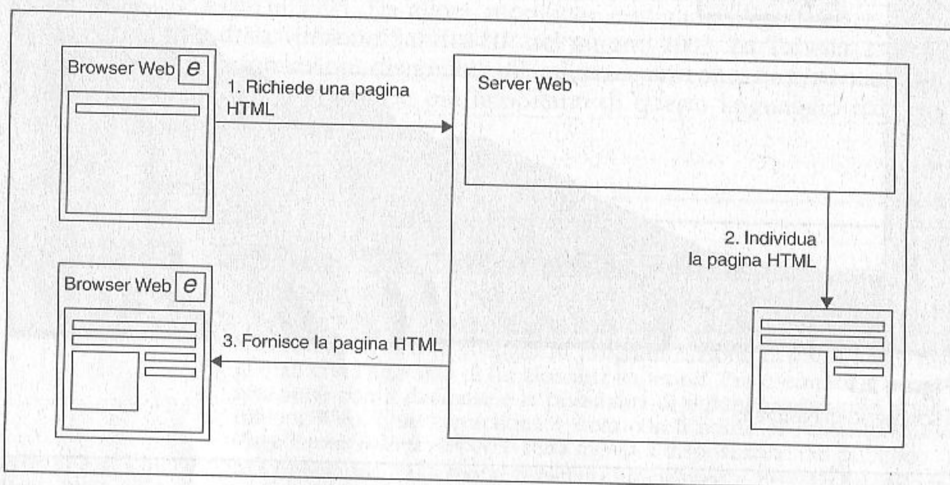


Figura 2.3
Passaggio di una pagina HTML.

Agli albori del Web fece la sua comparsa uno standard ideato proprio per la realizzazione di questi programmi. Tale standard, noto con il nome di *Common Gateway Interface* (i programmi conformi allo standard spesso prendono il nome di script CGI), specifica in che modo un server Web passa i dati a un programma in esecuzione. Gli script CGI possono essere scritti con una vasta gamma di linguaggi, come Perl, C, TCL e così via.
Ma come funzionano gli script CGI? Prendiamo in considerazione il diagramma della Figura 2.4.

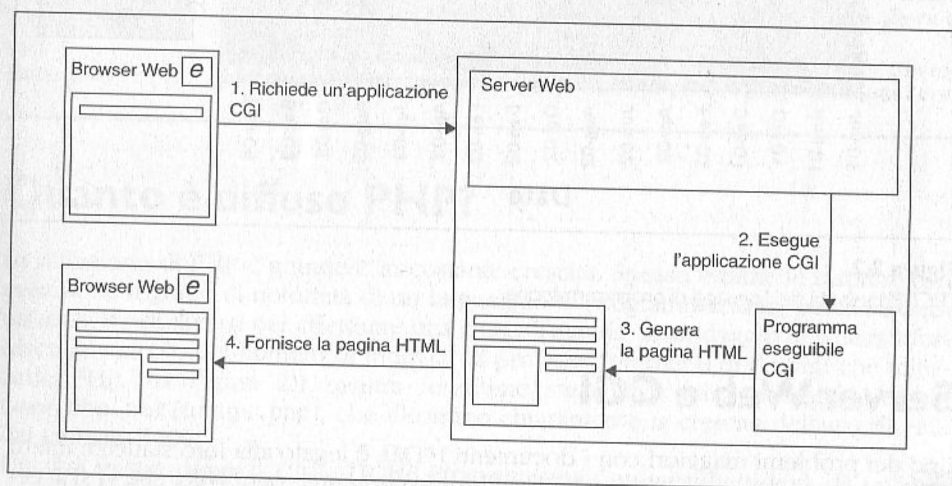


Figura 2.4
Creazione di una pagina Web dinamica con CGI.

In questo caso, l'elaborazione CGI inizia nel momento in cui un browser richiede un documento (nell'esempio si tratta di un'applicazione CGI) a un server Web. Al browser non importa se la pagina richiesta è un documento HTML statico o uno script CGI, poiché si limita a inviare la richiesta al server Web.
Quest'ultimo riconosce la richiesta di uno script CGI ed esegue l'applicazione CGI specifica. Lo script CGI fornisce codice HTML standard. Questo risultato viene passato al browser in modo da essere visualizzato sotto forma di pagina Web.
Il problema principale delle applicazioni CGI è che le istruzioni HTML che controllano il formato delle informazioni dinamiche prodotte dalle applicazioni tendono a disperdersi nel codice del programma. Questo complica la produzione di pagine Web dinamiche che possano risultare interessanti e di semplice fruizione, oltre a influire sulla manutenzione.

Funzionamento di PHP

L'approccio PHP alle pagine Web dinamiche differisce per un aspetto molto importante: invece di avere i tag HTML incorporati nel codice del programma, uno script PHP incorpora il codice del programma in HTML. La Figura 2.5 mostra il funzionamento di uno script PHP.

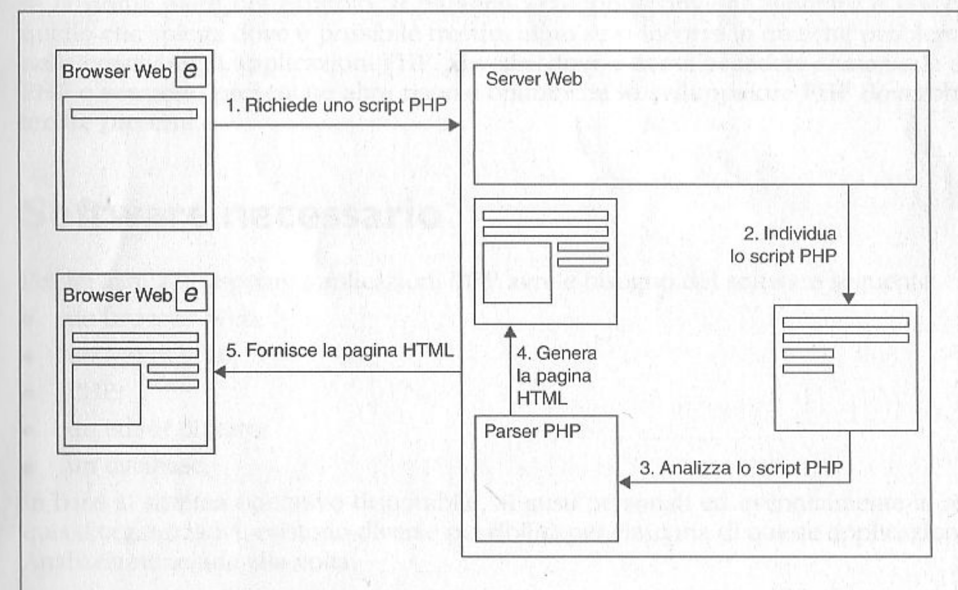


Figura 2.5
Creazione di una pagina Web dinamica con PHP.

Dalla Figura 2.5 potete vedere che quando un browser richiede uno script PHP, il server Web esegue il parser PHP e passa a tale interprete qualunque dato fornito dal browser, oltre ovviamente allo script PHP. Lo script PHP viene elaborato dal parser, il codice HTML contenuto nello script viene subito ripassato al browser, mentre il codice del programma viene eseguito e l'output ottenuto viene inviato al

browser. È importante considerare che è il programmatore PHP ad avere la responsabilità di garantire che il codice PHP fornisca sintassi HTML valida, poiché in caso contrario il browser non potrà visualizzarla correttamente.

Riepilogo

In questo capitolo abbiamo introdotto il linguaggio PHP e descritto la storia del suo sviluppo. Abbiamo mostrato che PHP è ampiamente utilizzato e che il suo impiego è ancora in una fase di crescita. Nel prossimo capitolo vedremo come e dove è possibile procurarsi una copia di PHP e quali siano le applicazioni associate necessarie per creare il proprio ambiente di sviluppo PHP.

Capitolo 3

Installazione dell'ambiente, aiuto e assistenza

Introduzione

Questo capitolo illustra dove è possibile reperire il software necessario per creare il proprio ambiente di sviluppo PHP. Se il provider di servizio o l'amministratore Web ha già installato un server Web, PHP e le applicazioni associate, potete saltare la prima parte del capitolo. Il paragrafo che non conviene ignorare è invece quello che spiega dove è possibile trovare aiuto se si incorre in qualche problema nella creazione di applicazioni PHP. Si vedrà dove e come accedere al manuale di PHP e verranno presentate altre risorse online che lo sviluppatore PHP dovrebbe tenere presenti.

Software necessario

Per iniziare a sviluppare applicazioni PHP avrete bisogno del software seguente:

- un browser Web;
- un server Web;
- PHP;
- un editor di testo;
- un database.

In base al sistema operativo disponibile, ai gusti personali ed eventualmente ai requisiti organizzativi, esistono diverse possibilità per ciascuna di queste applicazioni. Analizziamone una alla volta.

Browser Web

Il browser Web è l'applicazione utilizzata per visualizzare l'output dagli script HTML/PHP. I due browser più comuni sono Microsoft Internet Explorer e Netscape. Per gli utenti Microsoft Windows il più facile da ottenere è Internet Explorer, in quanto è incluso come parte integrante nel sistema operativo Windows. Anche Netscape è un browser conosciuto, soprattutto dagli utenti UNIX. Entrambi i browser sono costantemente aggiornati e migliorati. Potete procurarvi l'ultima versione di Internet

Explorer al sito www.microsoft.com/windows/ie/ e di Netscape dall'indirizzo home.netscape.com/it/download/.

Server Web

Il secondo elemento software necessario è un server Web. Un server Web riceve le richieste di documenti Web provenienti dai client Web (i browser), ottiene i documenti richiesti e li passa al browser Web affinché vengano visualizzati. Benché possiate sviluppare script HTML in un computer indipendente privo di server Web, questo non è possibile se desiderate utilizzare PHP.
I server Web disponibili per i vari sistemi operativi sono molti; alcuni sono prodotti commerciali potenti e costosi, mentre altri sono gratuiti, pur essendo prodotti di alta qualità sviluppati a livello professionale.
Per il sistema UNIX consigliamo il server Web Apache, disponibile all'indirizzo http://httpd.apache.org/download.cgi. Per le piattaforme Windows o Microsoft raccomandiamo il server Internet Information Services, incluso in Windows XP e Windows 2000, ma prelevabile anche dalla pagina www.microsoft.com/downloads/. Consigliamo inoltre Appserv, disponibile all'indirizzo http://appserv.sourceforge.net/.
Dato che ciascun server è diverso, dovrete leggere e seguire le istruzioni di installazione per ogni singolo prodotto.

PHP

Per visualizzare correttamente gli script PHP dovrete procurarvi una copia di PHP dal sito www.php.net. Dovrete scaricare la versione corretta di PHP per il vostro sistema operativo e configurare PHP e il server Web in modo che funzionino insieme. La maggior parte dei server Web fornisce le istruzioni necessarie in merito. Tuttavia non preoccupatevi e continuate a leggere, poiché tra poco sarà proposta una soluzione più semplice.

Editor di testo

Per cominciare a creare script HTML e PHP è necessario disporre di un editor di testo per modificare gli script. Qualunque editor di testo va bene e, naturalmente, ognuno ha le proprie preferenze. Emacs e vi sono particolarmente adatti per gli utenti UNIX, mentre Windows fornisce Blocco note, un editor di testo molto semplice ma non altrettanto potente. Per tutti gli esempi del libro abbiamo usato Blocco note, tuttavia potete senza alcun problema utilizzare un editor di testo che conoscete meglio.

Database - MySQL

L'ultimo componente necessario è un database. PHP interagisce con molti database diversi (per l'elenco completo fate riferimento alla documentazione di PHP), tuttavia abbiamo deciso di ricorrere al database MySQL poiché è gratuito e molto potente. MySQL è open source e può essere scaricato da http://www.mysql.com/ sia per le piattaforme Unix sia per quelle Windows.
MySQL ha un'interfaccia utente abbastanza complessa. Tuttavia, come vedremo nel prosieguo del libro, la guida è molto comoda grazie allo strumento PHPMyAdmin, che consente di configurare in modo rapido e facile i database MySQL. PHPMyAdmin è scaricabile da http://www.phpmyadmin.net/.

Bundle software integrati

I principianti di PHP possono incontrare una certa difficoltà nel districarsi tra tutti i software necessari. Come si può essere certi che tutti i componenti software siano installati e funzionino in modo corretto? Fortunatamente il problema è stato affrontato e la soluzione che ne è scaturita è quella del bundle di applicazioni. Per semplificare la vita dei principianti (e di buona parte degli utenti più esperti), le ultime versioni di tutto il software necessario per iniziare a sviluppare in PHP sono state raggruppate in un unico pacchetto e sono state configurate per funzionare correttamente insieme.
Si può scegliere tra una vasta gamma di bundle, molti dei quali sono elencati nel sito Web di HotScripts: http://www.hotscripts.com/PHP/Software_and_Servers/Installation_Kits/.
Per gli utenti Windows consigliamo il progetto Appserv Open, che dispone delle versioni più recenti del server AppServ e di PHP, MySQL e phpMyAdmin. Potete trovarlo all'indirizzo http://appserv.sourceforge.net/ ed è molto semplice da scaricare e installare.

Ottenere aiuto e assistenza

A prescindere dall'esperienza acquisita con PHP, prima o poi arriverà un momento in cui non saprete come comportarvi e avrete bisogno di aiuto e assistenza. In questi casi potreste trovare l'aiuto necessario nel manuale di PHP, oppure consultando altre fonti. Nei prossimi paragrafi del capitolo si vedrà dove conviene rivolgersi per ottenere aiuto e assistenza.

Leggete il manuale di PHP

Il manuale di PHP è disponibile online in diverse lingue (anche in italiano). Dalla pagina www.php.net/docs.php potete scegliere la lingua preferita e decidere se volete visualizzare una documentazione ricca di immagini o una più consona alle esigenze di stampa. La Figura 3.1 mostra un esempio del manuale online.
Chi desidera avere a disposizione una propria copia del libro, per poterlo consultare anche quando non è disponibile l'accesso a Internet, può scaricare il manuale in una vasta gamma di lingue e formati. La Tabella 3.1 elenca i vari tipi di documenti che è possibile scaricare.
Non tutti i formati elencati nella Tabella 3.1 sono disponibili in tutte le lingue. Tra tutti i manuali scaricabili disponibili, riteniamo particolarmente utile il formato di file .chm della guida di Windows. Il file fornisce un normale file di guida di Windows, che consente di cercare le informazioni in modo rapido e semplice e di vi-

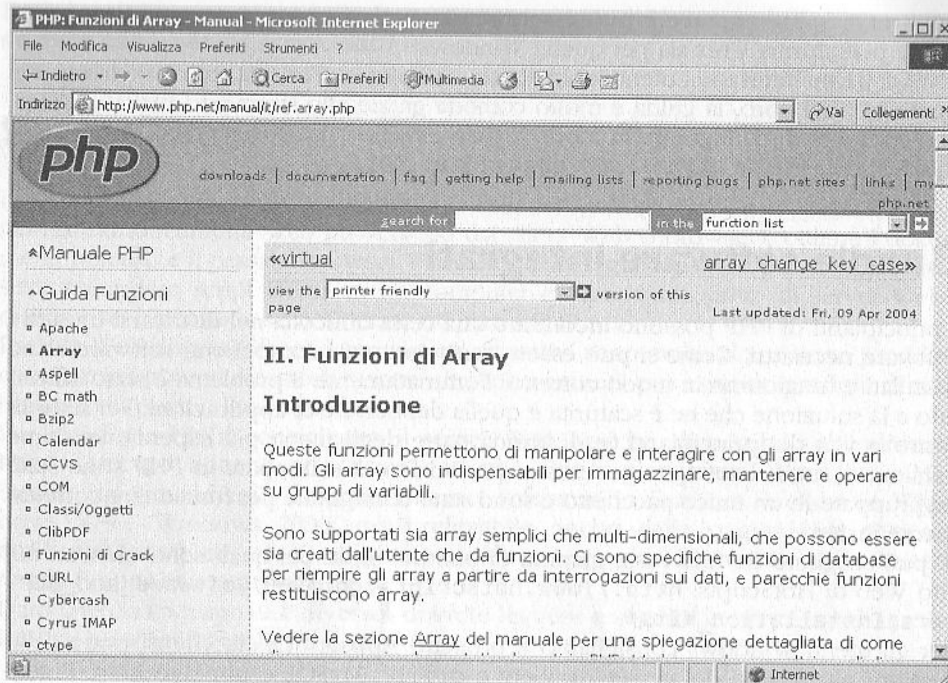


Figura 3.1
Il manuale PHP consultabile online.

Tabella 3.1 Formati di documento scaricabili

Tipo	Descrizione
HTML singolo	Un unico file HTML molto grande, contenente l'intero manuale.
HTML multiplo	Un file tar (compresso) composto da parecchi documenti HTML più piccoli, che costituiscono l'intero manuale.
PDF	Il manuale in formato PDF che può essere visualizzato con Adobe Acrobat. Questo file può essere installato su un palmare, purché sia disponibile una versione di Adobe Acrobat Reader per i palmari.
Palm Pilot DOC	Un file DOC standard per PalmPilot.
PalmPilot iSilo	Un file iSilo standard per PalmPilot.
Windows HTML Help	Un formato di file .chm per Windows.

sualizzarle in un formato facile e comodo da leggere. La Figura 3.2 mostra come si presenta il file quando si visualizza l'argomento delle strutture di controllo.

Manuale di MySQL

Il database MySQL, che utilizzeremo più avanti per creare applicazioni di database dinamiche, ha una documentazione completa, che si può scaricare. Tale documentazione è disponibile in diversi formati, simili a quelli del manuale di PHP. Le versioni scaricabili del manuale di MySQL sono disponibili alla pagina `http://dev.mysql.com/doc/`, riprodotta nella Figura 3.3.

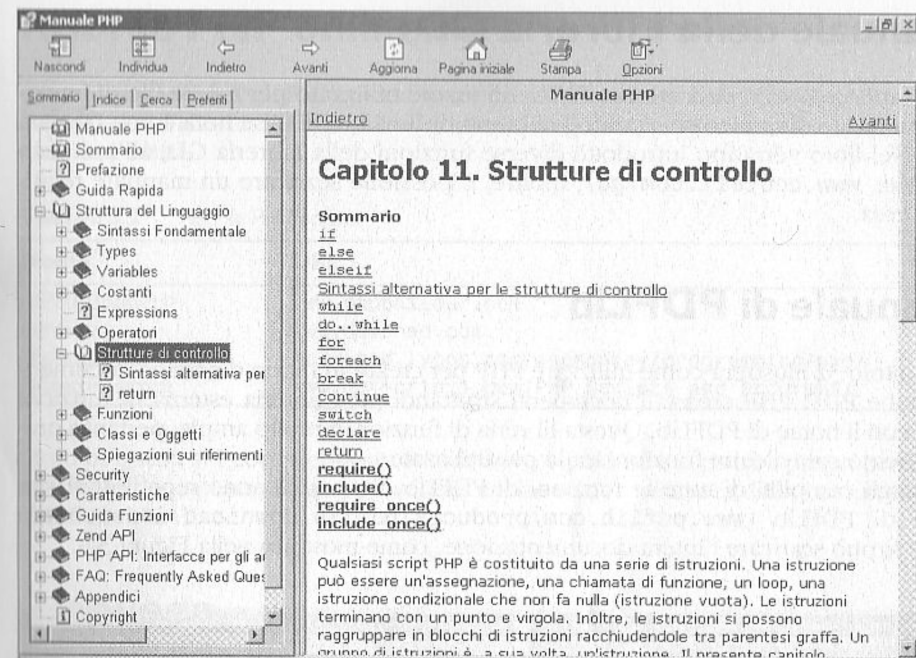


Figura 3.2
File della guida di Windows.

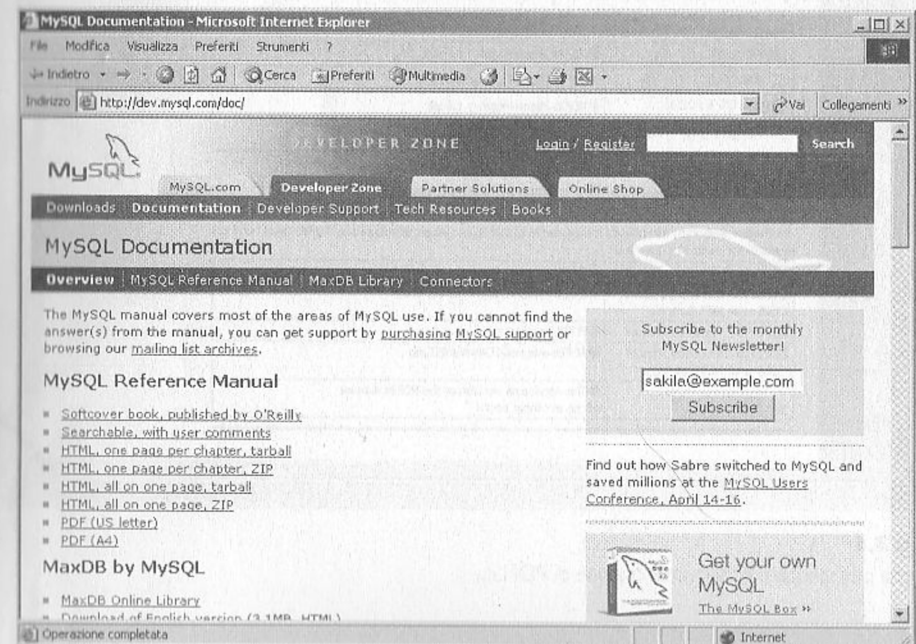


Figura 3.3
Pagina dei download di MySQL.

Manuale della Libreria GD

Nei Capitoli 30 e 31 vedrete che PHP può essere utilizzato per creare file di immagini in modo dinamico sfruttando la libreria di funzioni di terzi nota come Libreria GD. Nel libro verranno introdotte diverse funzioni della Libreria GD; all'indirizzo Internet `www.boutell.com/gd/`, inoltre, è possibile scaricare un manuale relativo a essa.

Manuale di PDFLib

Il Capitolo 32 illustrerà come utilizzare PHP per creare in modo dinamico documenti Adobe PDF. PHP crea tali documenti sfruttando una libreria esterna di funzioni nota con il nome di PDFLib. Questa libreria di funzioni è molto ampia, pertanto presenteremo solo alcune funzioni tra le più utilizzate.
I dettagli completi di tutte le funzioni di PDFLib disponibili sono reperibili al sito Web di PDFLib (`www.pdfiib.com/products/pdfiib/download/index.html`) dove si può scaricare l'intera documentazione, come mostrato nella Figura 3.4.

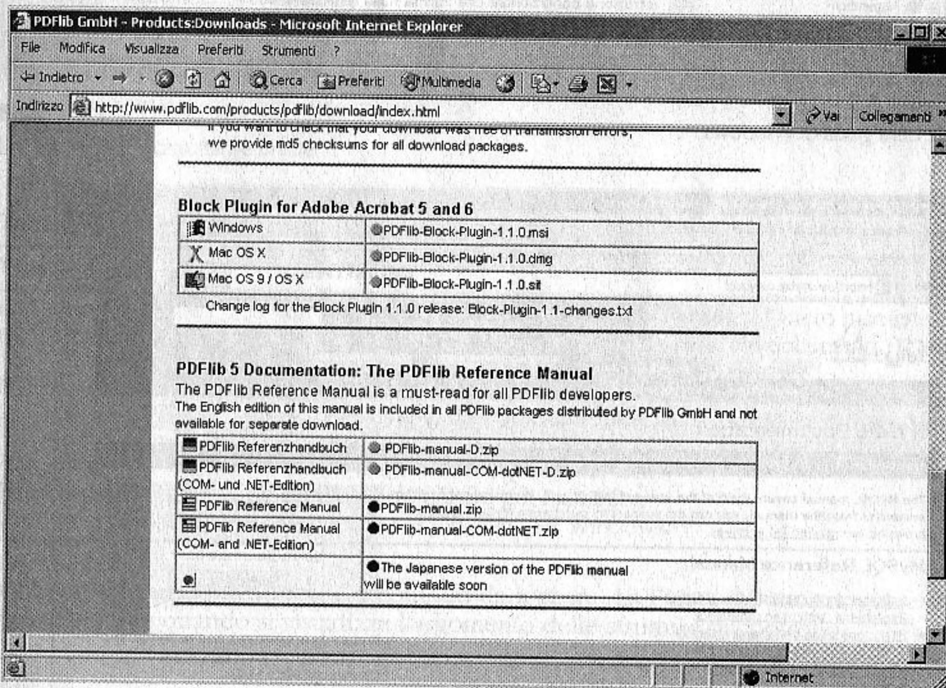


Figura 3.4
La pagina per scaricare la documentazione di PDFLib.

Risorse PHP online

Oltre al sito ufficiale di PHP, esistono molte utili comunità online create da e per gli sviluppatori PHP. La Tabella 3.2 offre un riepilogo di alcuni tra questi siti che meritano una visita.

Tabella 3.2 Risorse PHP online.

Nome	URL
<?PHPBuilder?>	`www.pnpbuilder.com`
DevShed	`www.devshed.com`
Webmonkey	`hotwired.lycos.com/webmonkey/programming/php/`
Hotscripts.com	`www.hotscripts.com/PHP/Scripts_and_Programs/`

Molti di questi siti comprendono forum dove è possibile esporre i propri problemi e ricevere le risposte (in genere valide, anche se non sempre) dalla comunità PHP del Web.

Riepilogo

In questo capitolo abbiamo spiegato quali sono i componenti software necessari per creare un ambiente di sviluppo PHP. Si è visto inoltre dove è possibile accedere al manuale di PHP, sia come risorsa online sia come manuale scaricabile. Abbiamo descritto dove è possibile ottenere il manuale per il database MySQL e per le varie librerie di funzioni che saranno utilizzate nel prosieguo del libro. Nel prossimo capitolo presenteremo il primo script PHP e descriveremo il ciclo vitale dello sviluppo del software PHP.

Capitolo 4

Il primo programma PHP

Introduzione

In questo capitolo verrà presentato un primo script PHP, oltre alle informazioni necessarie per creare, salvare ed eseguire gli script. Spiegheremo come vengono segnalati gli errori negli script e come dovete comportarvi quando vi imbattete in essi.

Un primo script PHP

Iniziamo esaminando un primo script PHP:

```
<?php
// Primo - Esempio 4-1
//--------------------

echo "Salve a tutti e benvenuti nel primo script PHP.";

?>
```

L'esempio è stato volutamente mantenuto il più semplice possibile, in quanto l'intento era mostrare come appare uno script PHP. Le righe dello script precedente saranno esaminate una per una.

```
<?php
```

Questa riga segna l'inizio di una serie di istruzioni PHP. Le due righe che iniziano con "//" sono commenti:

```
// Primo - Esempio 4-1
//--------------------
```

Volendo è possibile omettere queste righe, in quanto il loro unico scopo è fornire istruzioni e informazioni destinate a chi legge il codice, ma che vengono ignorate dal computer. La riga successiva inizia con l'istruzione `echo`:

```
echo "Salve a tutti e benvenuti nel primo script PHP.";
```

Questa istruzione comunica al parser di inviare alla pagina Web le informazioni contenute tra le doppie virgolette successive all'istruzione `echo`. Infine trovate la riga:

```
?>
```

Questa indica la fine delle istruzioni PHP. Le righe vuote dello script non hanno alcun significato e vengono incluse semplicemente per agevolare la lettura dello script.

Modifica degli script PHP

La prima operazione da compiere quando vi trovate di fronte a uno script PHP come il precedente è immetterlo nel computer e salvarlo. Gli script PHP vengono memorizzati come semplici file di testo e in quanto tali richiedono un editor di testo per poter essere immessi, modificati e salvati. Come già detto, potete utilizzare qualunque editor di testo (per questo libro è stato utilizzato Blocco note con Windows). La Figura 4.1 illustra lo script precedente inserito nel Blocco note di Windows.

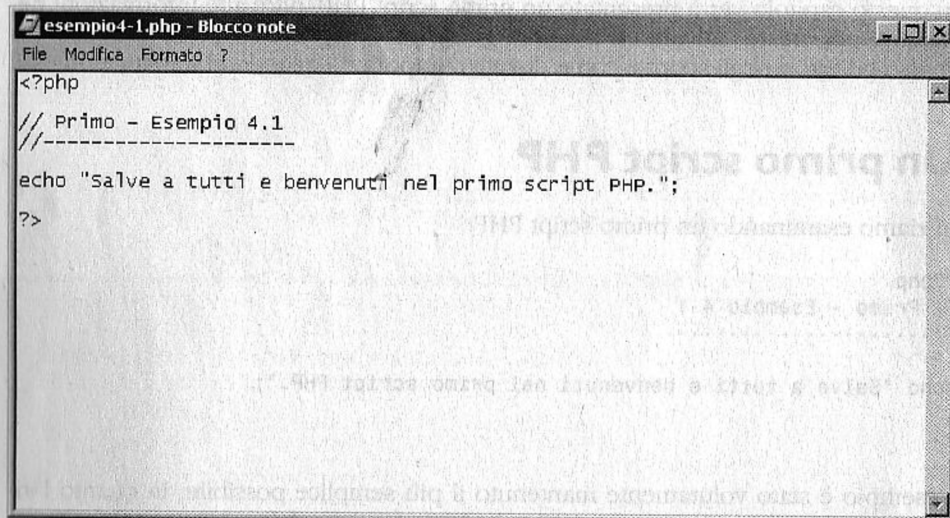


Figura 4.1
La finestra di Blocco note con uno script PHP al suo interno.

Salvataggio degli script PHP

La posizione esatta in cui vengono salvati gli script PHP dipenderà dall'ambiente di sviluppo utilizzato, e in particolare dal server Web installato. Sul computer utilizzato per questo libro era installato il server Web Internet Information Services per Windows, che di default crea la struttura di directory standard seguente sull'unità C:\ del PC:

C:
+ Inetpub
+ wwwroot

Tutti i documenti HTML e gli script PHP devono essere salvati nella directory *wwwroot* o in sottodirectory di quest'ultima. Nel nostro caso è stata creata la struttura di directory seguente sotto la directory *wwwroot*:

C:
+ Inetpub
+ wwwroot
+ phpbook
+ files
+ graphics
+ pdf
+ uploads

Tutti gli script PHP degli esempi verranno memorizzati nella directory *phpbook*, mentre le altre sottodirectory al di sotto di essa verranno impiegate nei capitoli successivi per memorizzare i file generati, le immagini, i documenti PDF e qualunque file che venga caricato. Per salvare lo script PHP nella directory *phpbook* con Blocco note occorre selezionare l'opzione *File/Salva con nome*, portarsi alla directory *phpbook* e inserire un nome di file per lo script, che deve terminare in .php. Abbiamo scelto di salvare il file come *esempio4-1.php*. Osservate che con Blocco note dovete accertarvi che l'opzione *Salva come* sia impostata su *Tutti i file*, altrimenti il programma aggiungerà l'estensione .txt ai file salvati e questi non funzioneranno correttamente. La Figura 4.2 illustra quanto appena detto.

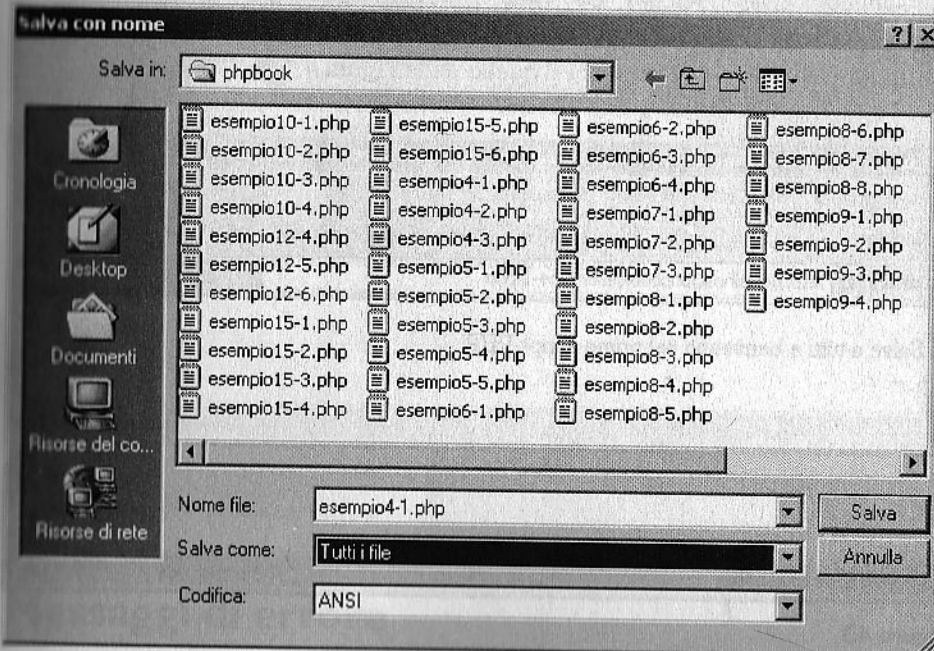


Figura 4.2
Salvataggio dei file in Blocco note.

Infine, per salvare il file dovete fare clic su *Salva*.

Visualizzazione dell'output PHP

Una volta creato e salvato lo script, potete visualizzare l'output che produce. A tal fine dovete procedere nel modo seguente.

1. Utilizzare un client Web (noto anche come browser Web). Per tutti gli esempi del libro verrà utilizzato Microsoft Internet Explorer, tuttavia, dato che PHP genera codice HTML, gli script presentati dovrebbero funzionare con qualunque browser Web.
2. Accertarvi che il server Web sia in esecuzione, in quanto sarà necessario per elaborare gli script PHP.

Per visualizzare l'output dello script precedente dovete digitare, nel campo degli indirizzi del browser, l'indirizzo seguente:

```
http://localhost/phpbook/esempio4-1.php
```

Questo indirizzo Web è di tipo "locale" e viene utilizzato solo a fini di sviluppo, quando il server Web impiegato si trova sul medesimo computer sul quale avviene lo sviluppo degli script. Esaminiamo il significato di questo indirizzo:

- `http://` indica che si tratta di un indirizzo Web *HyperText Transfer Protocol*;
- `localhost` è il nome del server Web: come si è detto, si tratta di un indirizzo utilizzato a fini di sviluppo;
- `/phpbook` è la directory in cui sono memorizzati gli script;
- `/esempio4-1.php` è il nome dello script.

L'output prodotto da questo script è quello della Figura 4.3.

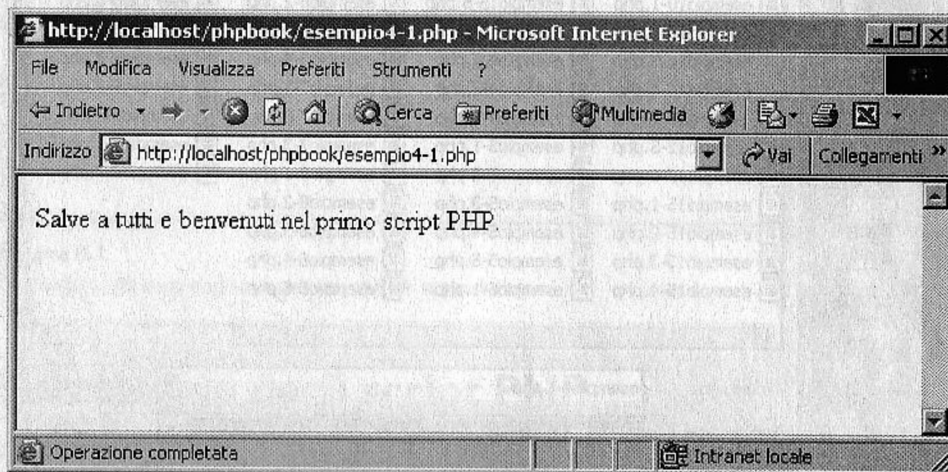


Figura 4.3
Il primo script PHP.

Un secondo script PHP

Provate ora con un altro script un po' più complesso:

```
<?php

// Primo - Esempio 4-2
//---------------------


echo "Oggi è il " . showDate();

function showDate() {
      $date = getthedate();
      return ($date["year"] .'-'. $date["mon"] . '-' $date["mday"]);
}
```

Al momento non spiegheremo il funzionamento dello script, i cui diversi componenti saranno presentati e analizzati nei capitoli successivi. Esso infatti è stato realizzato solo per illustrare come PHP possa generare una semplice pagina Web dinamica. Inserite e salvate lo script precedente nella directory del server Web e visualizzate lo script risultante nel browser Web. L'output prodotto è quello della Figura 4.4.

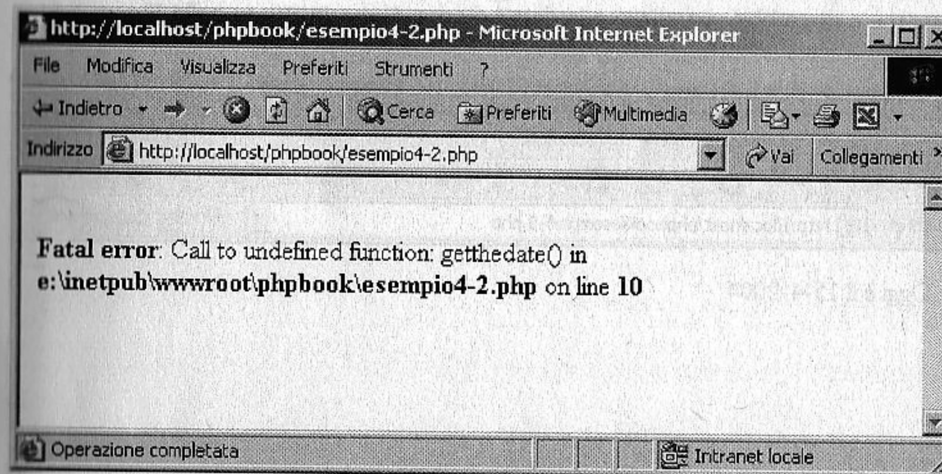


Figura 4.4
Il secondo script PHP produce un errore.

Messaggi di errore

A quanto pare c'è un problema, poiché il secondo script PHP ha prodotto un messaggio di errore.

```
Fatal error: Call to undefined function: getthedate() in
e:\inetpub\wwwroot\phpbook\esempio4-2.php on line 10
```

In realtà si tratta di un errore voluto, per mostrare come appare un messaggio di errore e cosa occorre fare per correggerlo. Il messaggio di errore implica che esiste un problema relativo alla riga:

```
$date = getthedate();
```

In realtà infatti essa dovrebbe essere scritta nel modo seguente:

```
$date = getdate();
```

Ecco quindi lo script corretto:

```
<?php
// Primo - Esempio 4-3
//---------------------
echo "Oggi è il " . showDate();
function showDate() {
	$date = getdate();
	return ($date["mday"] . '-'. $date["mon"] . '-'. $date["year"]);
}
```

Correggete lo script e salvatelo nuovamente. Se lo salvate con lo stesso nome di quello contenente l'errore, fate semplicemente clic sul pulsante *Aggiorna* del browser; se invece avete scelto un nuovo nome, digitate nuovamente l'indirizzo nel browser e premete il tasto Invio. Lo script questa volta dovrebbe funzionare correttamente e dovrebbe visualizzare la data odierna, come mostrato nella Figura 4.5.

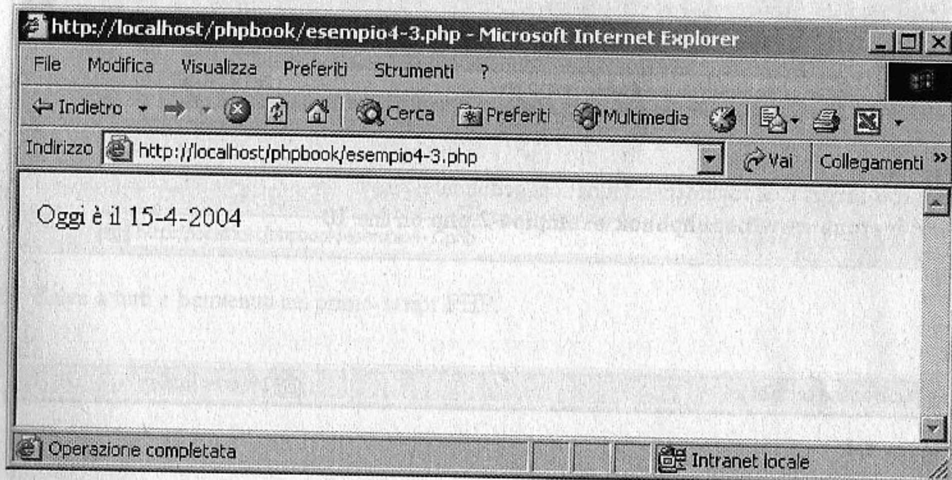


Figura 4.5
Il secondo script PHP.

Lo script presentato è di tipo dinamico e visualizzerà la data corretta ogni volta che decidete di visualizzarne l'output.

Riepilogo

In questo capitolo sono stati presentati i primi script PHP assieme a una spiegazione di come immetterli e salvarli sul proprio computer. Abbiamo spiegato come sia possibile visualizzare l'output degli script e come appaiono i messaggi di errore quando si verificano problemi. Il prossimo capitolo proseguirà nell'esame di PHP analizzando in modo più approfondito alcuni elementi fondamentali del linguaggio.

Capitolo 5

Nozioni fondamentali di PHP

Introduzione

Questo capitolo prende in esame in modo più approfondito alcuni aspetti fondamentali dello sviluppo con il linguaggio PHP. Si vedrà com'è possibile incorporare gli script PHP nei documenti HTML, come formattare gli script, il ruolo dei commenti e l'istruzione `echo`.

Dentro e fuori da PHP

In precedenza si è visto che uno script PHP deve iniziare con l'istruzione `<?php` e terminare con `?>`. Quindi, per esempio, il primo script PHP è stato illustrato nel modo seguente:

```
<?php

//Nozioni fondamentali - Esempio 5-1
//---------------------

echo "Salve a tutti e benvenuti nel primo script PHP.";

?>
```

Tuttavia gli script PHP possono essere incorporati nei documenti HTML generando una mescolanza di codice HTML e PHP. Considerate l'esempio che segue:

```
<body>
Salve, questo è codice HTML standard

<?php

// Nozioni fondamentali - Esempio 5-2
//---------------------

echo " mentre questo è generato da PHP.";

?>
</body>
```

L'esempio precedente inizia con un tag HTML `<BODY>` seguito da una stringa di testo, "Salve, questo è codice HTML standard". Poi trovate uno script PHP che utilizza il comando echo per visualizzare il testo "mentre questo è generato da PHP". Dopo l'elemento di chiusura ?> di PHP si ritorna al codice HTML con un tag body di chiusura. Il fatto interessante è la possibilità di entrare e uscire da PHP tutte le volte che si vuole in un documento HTML. Considerate l'esempio seguente:

```
<body>
Salve, questo è codice HTML standard

<?php

// Nozioni fondamentali - Esempio 5-3
//--------------------

echo " mentre questo è prodotto con PHP.";
?>

Ora eccoci di nuovo in HTML

<?php
echo " mentre questo è ancora una volta PHP.";
?>

Per finire, utilizziamo ancora HTML.

</body>
```

Questo esempio illustra che non siete limitati a un solo script `<?php ?>`, ma che potete includerne diversi. L'output prodotto dallo script precedente è quello della Figura 5.1.

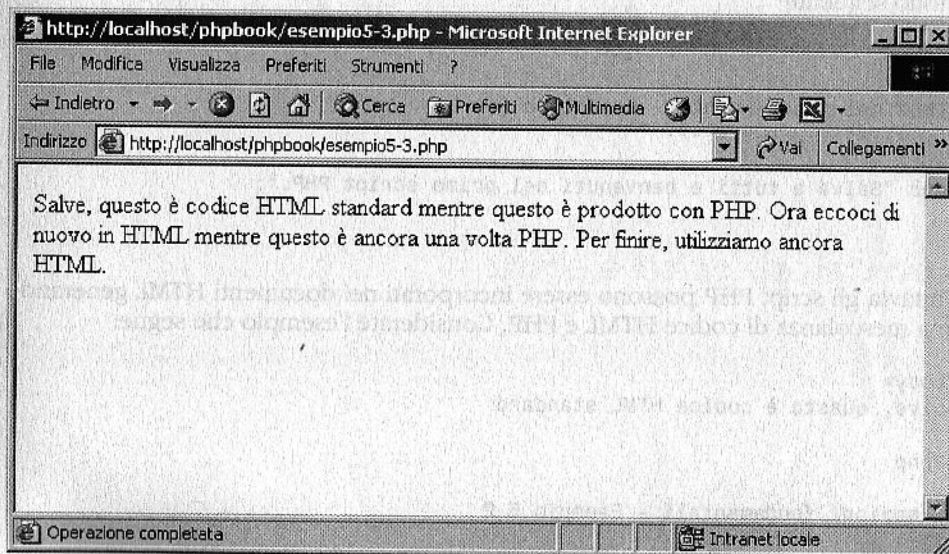


Figura 5.1
Dentro e fuori da PHP.

Formattazione delle istruzioni PHP

Il linguaggio di scripting PHP è piuttosto tollerante sulla disposizione delle proprie istruzioni. Per esempio, queste righe:

```
<?php
echo "Ciao a tutti";
?>
```

sono un modo perfettamente accettabile di includere istruzioni PHP in un documento HTML; anche la riga seguente, però, è altrettanto valida:

```
<?php echo "Ciao a tutti" ?>
```

Osservate che nel secondo esempio manca il punto e virgola (;) alla fine dell'istruzione echo. Esso viene utilizzato per terminare un'istruzione PHP, proprio come avviene con il tag ?>; pertanto può essere omesso quando il codice PHP è scritto su una sola riga.
Va detto, però, che la prima forma in genere è di più agevole comprensione, ragione per cui tutti gli esempi del libro saranno presentati con questo stile. Se volete, potete rendere totalmente illeggibili gli script collocando molte istruzioni separate sulla stessa riga. Per esempio:

```
<?php

// Nozioni fondamentali - Esempio 5-4
//--------------------
echo "Oggi è il " . showDate(); function showDate()
  {$date = getdate();
return ($date["year"] . '-'. $date["mon"] . '-' . $date["mday"]);}
?>
```

Quella appena presentata è una riscrittura dello script *esempio4-3.php* per la visualizzazione della data presentato nel capitolo precedente. Si consiglia di non trascurare la leggibilità degli script, in quanto vi accorgerete che essa agevola la correzione degli errori, l'introduzione di modifiche o la semplice comprensione del funzionamento di uno script.

Istruzioni echo e print

echo è un costrutto del linguaggio PHP che consente di visualizzare stringhe di testo sulla pagina Web.
Abbiamo già usato echo per visualizzare il testo negli script di esempio precedenti, tuttavia ecco un altro esempio di questa istruzione:

```
echo "Ciao";
```

Questa istruzione echo visualizza il testo "Ciao" sulla pagina Web. Le virgolette doppie (") indicano l'inizio e la fine del testo da visualizzare. In questo esempio il testo "Ciao" è molto breve, tuttavia se aveste una quantità di testo molto maggiore

da visualizzare, potreste suddividere le stringhe di esempio su più righe, come in questo esempio:

```
echo "Ciao, questa stringa
occupa diverse
righe";
```

Questo script visualizza il testo "`Ciao, questa stringa occupa diverse righe`", che verrà visualizzato su una sola riga, non suddiviso su più righe come potreste immaginare. Potete provare voi stessi visualizzando lo script seguente:

```
<?php
// Nozioni fondamentali - Esempio 5-5
//--------------------

echo "Ciao, questa stringa
occupa diverse
righe";
?>
```

L'output è illustrato nella Figura 5.2.

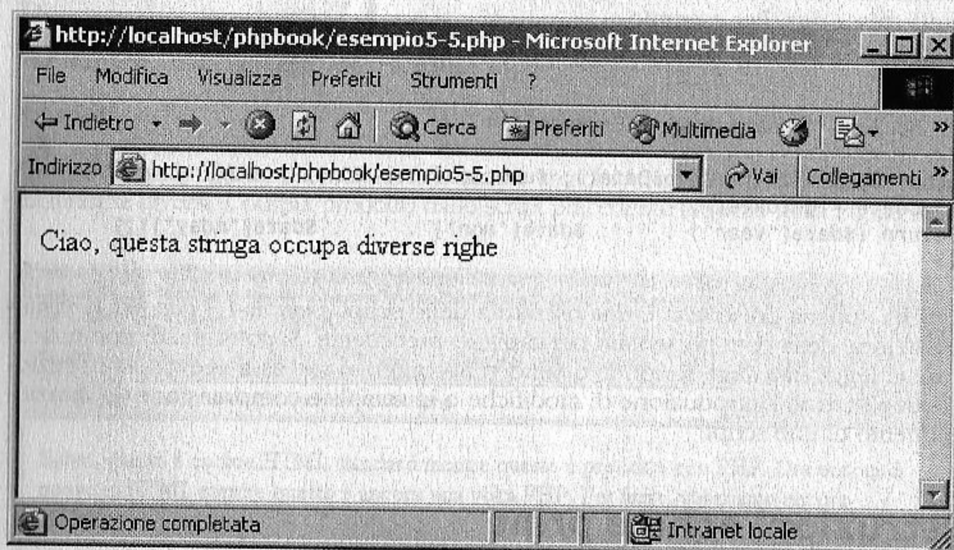


Figura 5.2
Istruzione echo su più righe.

L'istruzione `echo` consente di visualizzare variabili e più stringhe di caratteri, come si vedrà nei capitoli successivi. PHP supporta anche un'altra istruzione per agevolare l'output, `print`, che funziona esattamente come `echo`:

```
print "Salve a tutti";
```


Questa istruzione visualizzerà il testo "Salve a tutti". Nei vostri script potete utilizzare indifferentemente `echo` o `print`.
Tutti gli esempi del libro utilizzano `echo`.

Commenti

PHP supporta tre forme di commenti, che possono essere inclusi per aiutare chi legge il codice a comprendere il funzionamento dello script PHP. Il primo tipo di commento occupa più righe ed è uguale a quello introdotto dal linguaggio di programmazione C:

```
/* Questo è un commento
che può essere separato in
più righe */
```

I caratteri `/*` indicano l'inizio del commento, che si chiude in corrispondenza dei caratteri `*/`. La seconda forma di commento è a riga singola ed è uguale a quello introdotto dal linguaggio di programmazione C++:

```
// commento a riga singola.
```

I caratteri `//` indicano l'inizio di un commento a riga singola. Infine, la terza forma dei commenti PHP è analoga a quelli della shell UNIX e anch'essa occupa una sola riga:

```
# Un altro commento a riga singola.
```

I commenti vengono ignorati dal parser PHP e sono di aiuto solo per chi legge il codice, in quanto spiegano cosa sta accadendo. Tutti gli script completi di questo libro iniziano con due righe di commento simile alle seguenti:

```
// Nozioni fondamentali - Esempio 5-4
//--------------------
```

Questo commento fornisce il nome dello script e indica come dovrebbe essere salvato. Nell'esempio precedente, dato che il nome è `Esempio 5-4`, dev'essere salvato come *esempio5-4.php*. Il nome dello script consente di sapere che si tratta dell'Esempio 4 del Capitolo 5.
Negli esempi i commenti sono stati evitati il più possibile e, quando presenti, si è cercato di ottenere la massima brevità e semplicità.

Riepilogo

In questo capitolo abbiamo visto come si costruisce uno script PHP e i mezzi che consentono di incorporarlo in un documento HTML. Abbiamo esaminato l'istruzione `echo` e il ruolo dei commenti nel linguaggio. Il prossimo capitolo proseguirà l'analisi del linguaggio PHP e presenterà le variabili.

Parte II

Costrutti fondamentali del linguaggio

Capitolo 6

Introduzione alle variabili

Introduzione

In questo capitolo verrà introdotto il concetto di variabili, esaminando come vengono definite e utilizzate in PHP, le limitazioni al loro impiego e la loro grande importanza. Per cominciare spiegheremo che cosa sono le variabili.

Che cos'è una variabile?

Le variabili sono contenitori dotati di nome, in grado di contenere valori. Il valore detenuto da una variabile può cambiare durante l'esecuzione dello script, da cui il nome di "variabile". Le variabili possono contenere numeri, caratteri o stringhe che normalmente rappresentano qualcosa, come per esempio date, cognomi o salari.
Tutte le variabili hanno un nome univoco, che consente di fare riferimento a esse in modo non ambiguo.

Variabili in PHP

Le variabili in PHP sono definite dal simbolo di dollaro ($) seguito dal nome della variabile. Il nome distingue tra maiuscolo e minuscolo e pertanto `$SIMON` è diverso da `$simon`. I nomi delle variabili devono iniziare con una lettera o un carattere di sottolineatura, che può essere seguito da un numero qualunque di lettere, numeri o caratteri di sottolineatura.
Ecco per esempio alcuni nomi di variabile validi:

```
$name;
$chapter45;
$_var;
```

Quello che segue è invece un nome di variabile non accettabile, in quanto non inizia con una lettera o un carattere di sottolineatura.

```
$1stName;
```

Assegnare a variabili per valore

Fino alla comparsa di PHP 4, l'assegnazione per valore era l'unico modo per attribuire valori a una variabile. Quando il valore di una variabile sorgente viene assegnato in questo modo a una variabile di destinazione, esso viene copiato nella variabile di destinazione; pertanto, dato che viene effettuata una copia del valore, la modifica del valore della variabile originale non influirà in alcun modo sul valore della variabile di destinazione. Per esempio, supponiamo di avere due variabili denominate `$tax` e `$net`, alle quali sono stati assegnati i valori 17 e 200. Se a questo punto dichiarassimo una variabile denominata `$copyOfNet` e le assegnassimo il valore della variabile `$net`, in caso di modifiche a quest'ultima, la variabile `$copyOfNet` non verrebbe influenzata. La Figura 6.1 illustra quanto appena detto.

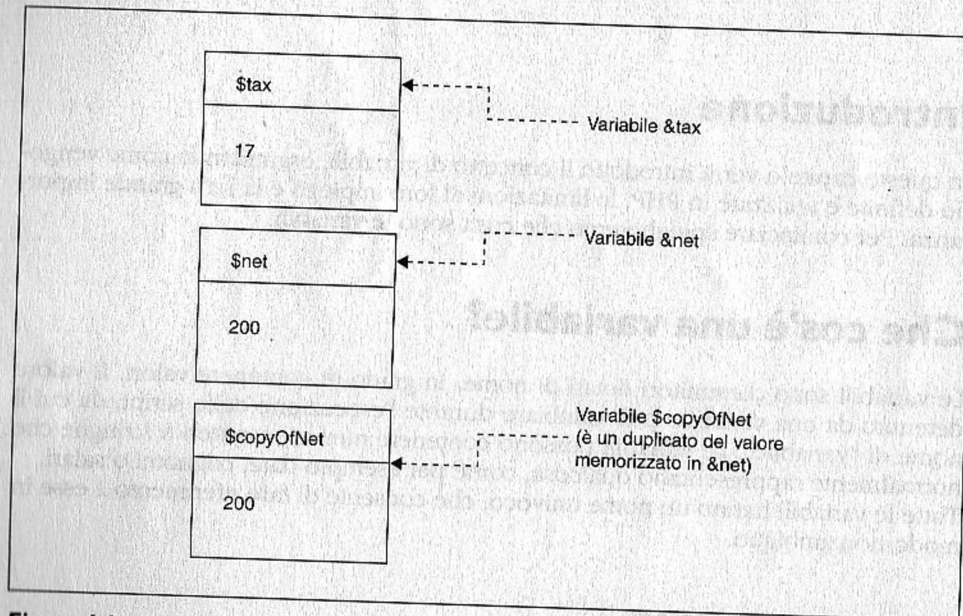


Figura 6.1
Assegnare a una variabile per valore.

Per assegnare un valore a un variabile si utilizza il carattere di uguale (=). Lo script seguente illustra l'assegnamento ad alcune variabili per valore:

```
<?php
// Variabili - Esempio 6-1
//--------------------

$tax = 17;
$net = 200;
$copyOfNet = $net;

?>
```

L'esempio precedente illustra la definizione di tre variabili: `$tax`, cui viene assegnato il valore 17, `$net`, che ha valore 200, e `$copyOfNet`, cui viene assegnato il valore

di `$net`. Tutto questo mostra che alle variabili è possibile assegnare non solo costanti, ma anche i valori di altre variabili.

Assegnare a variabili per riferimento

L'introduzione di PHP 4 ha portato con sé un altro modo di assegnare valori alle variabili, l'assegnamento per riferimento, grazie al quale la nuova variabile diviene un semplice riferimento, noto anche come alias, alla variabile originale; a volte viene utilizzata l'espressione di "puntamento alla" variabile. La Figura 6.2 illustra quanto appena detto.

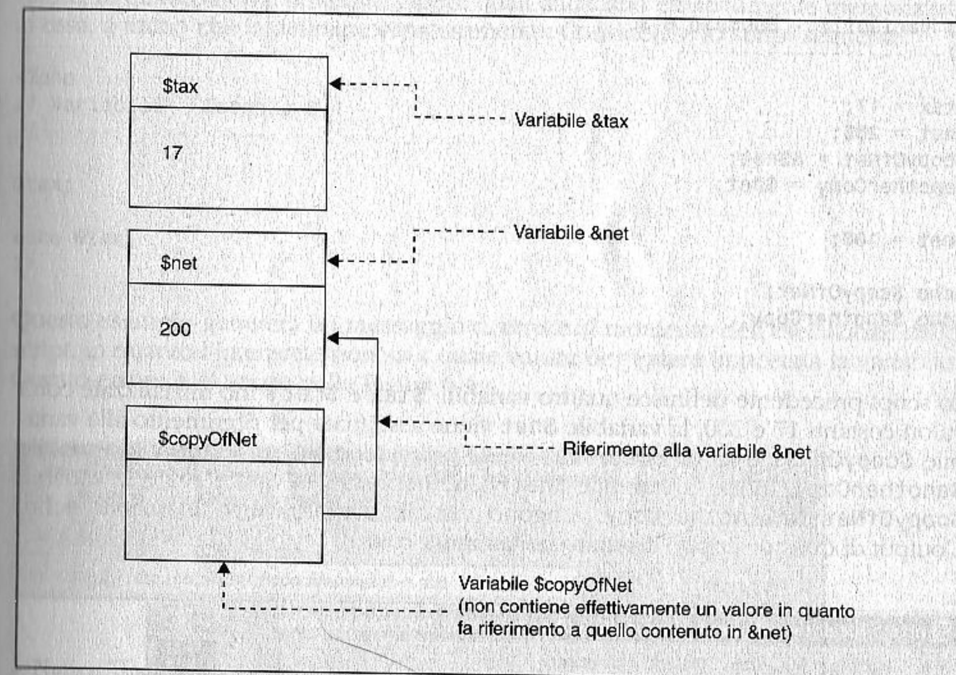


Figura 6.2
Assegnare a una variabile per riferimento.

In questo caso una modifica alla variabile originale o a quella nuova influisce su entrambe. Dato infatti che non viene effettuata alcuna copia, questa forma di assegnamento può portare a un aumento delle prestazioni, anche se non è così facile rendersene conto. Oltre a ciò, osservate che l'assegnamento a una variabile per riferimento non consente di assegnare una costante. Per ottenere un assegnamento per riferimento, dovete semplicemente aggiungere una e commerciale (&) all'inizio della variabile che viene assegnata. Lo script dell'esempio seguente illustra l'assegnamento per riferimento:

```
<?php
// Variabili - Esempio 6-2
//--------------------
```

```
$tax = 17;
$net = 200;
$copyOfNet = &$net;
?>
```

Visualizzazione delle variabili

Le variabili possono essere visualizzate con l'istruzione `echo`; considerate l'esempio che segue:

```
<?php

// Variabili - Esempio 6-3
//--------------------

$tax = 17;
$net = 200;
$copyOfNet = &$net;
$anotherCopy = $net;

$net = 100;

echo $copyOfNet;
echo $anotherCopy;
?>
```

Lo script precedente definisce quattro variabili: `$tax` e `$net` sono inizializzate con i valori costanti 17 e 200, la variabile `$net` viene assegnata per riferimento alla variabile `$copyOfNet` e la variabile `$net` viene poi assegnata per valore alla variabile `$anotherCopy`. Infine la variabile `$net` viene impostata al valore 100 e le variabili `$copyOfNet` e `$anotherCopy` vengono visualizzate attraverso istruzioni `echo`. L'output di questo script è illustrato nella Figura 6.3.

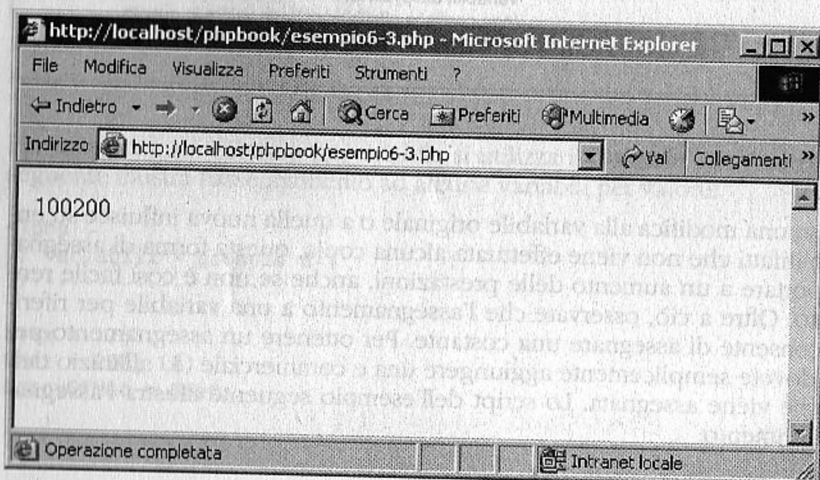


Figura 6.3
Output delle variabili.

L'output prodotto da questo script, anche se non è presentato in modo elegante, illustra che la variabile che riceve l'assegnamento per riferimento cambia il proprio valore da 200 a 100 quando la copia originale (`$net`) viene alterata. La variabile `$anotherCopy` invece, che ha ricevuto un assegnamento per valore, mantiene il valore originale 200.

Dichiarazione di variabili senza assegnamento di valori

Non è consigliabile dichiarare una variabile e poi utilizzarla prima di assegnarle un valore, in quanto non è possibile sapere quali dati siano effettivamente memorizzati in essa, a meno che la definiate espressamente. Considerate lo script seguente:

```
<?php
// Variabili - Esempio 6-4
//--------------------

$tax;

echo $tax;
?>
```

Questo esempio genererà un messaggio di errore al momento dell'esecuzione dello script, in quanto l'interprete non sa a quale valore dev'essere impostata la variabile. Questo errore è illustrato nella Figura 6.4.

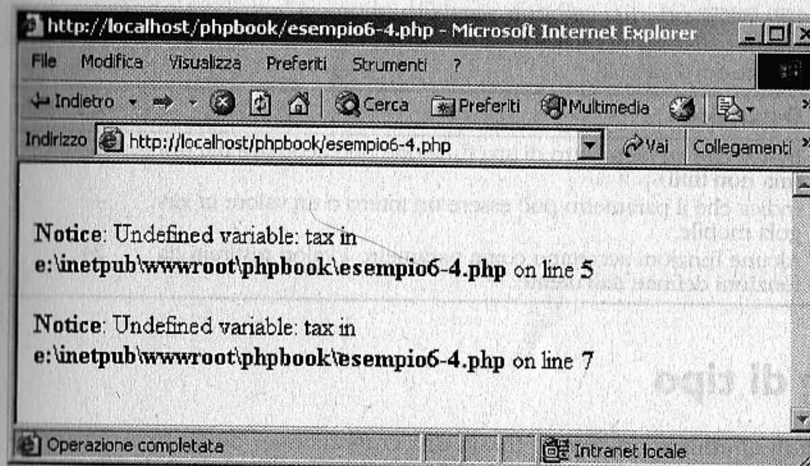


Figura 6.4
Avviso di variabile indefinita.

Tipi di variabili

Un tipo è la descrizione del formato in cui le informazioni sono memorizzate in una variabile. PHP riconosce otto tipi primitivi, che sono elencati nella Tabella 6.1 insieme a una breve descrizione e all'indicazione del capitolo nel quale sono introdotti e descritti per la prima volta.

Tabella 6.1 Tipi di variabili

Tipo	Descrizione	Capitolo
`Boolean`	Una variabile booleana esprime un valore di verità, che può essere `TRUE` o `FALSE`.	7
`Integer`	Una variabile intera è un numero appartenente all'insieme Z = {..., −2, −1, 0, 1, 2, ...}.	7
`float`, `double`	Numeri in formato a virgola mobile (anche detti "numeri reali").	7
`String`	Una stringa è una serie di caratteri. In PHP un carattere equivale a un byte.	8
`Array`	Sequenza indicizzata di una o più variabili.	14
`Object`	Variabile speciale definita dall'utente, nell'ambito dalle programmazione orientata agli oggetti.	37
`Resource`	Una risorsa è una variabile speciale che contiene un riferimento a una risorsa esterna, per esempio a un database.	36
`NULL`	NULL rappresenta una variabile priva di valore.	7

Il manuale di PHP introduce inoltre alcuni pseudo-tipi per ragioni di leggibilità. Questi pseudo-tipi, elencati nella Tabella 6.2, vengono utilizzati principalmente per aiutare a descrivere i prototipi delle funzioni; più avanti li potrete vedere impiegati proprio in questa veste.

Tabella 6.2 Pseudo-tipi di variabili

Tipo	Descrizione	Capitolo
`Mixed`	Indica che un parametro di una funzione può accettare più tipi (ma non tutti).	17
`Number`	Indica che il parametro può essere un intero o un valore in virgola mobile.	–
`Callback`	Alcune funzioni accettano come parametro i valori restituiti da funzioni definite dall'utente.	24

Casting di tipo

PHP consente di effettuare un casting (ossia una conversione) da un tipo di variabile a un altro. Per specificare un casting è sufficiente collocare il tipo desiderato tra parentesi prima della variabile che dev'essere convertita. Per esempio:

```
$var = 4;
$bool = (Boolean) $var;
```

Nell'esempio precedente `$var` è un intero e viene convertito a `$bool` come valore booleano. Le diverse operazioni di casting consentite sono elencate nella Tabella 6.3.

Tabella 6.3 Casting

Casting	Descrizione
`(int)`, `(integer)`	Casting a intero.
`(bool)`, `(Boolean)`	Casting a booleano.
`(float)`, `(double)`, `(real)`	Casting a un tipo in virgola mobile.
`(string)`	Casting a stringa.
`(array)`	Casting ad array.
`(object)`	Casting a oggetto.

Ambito delle variabili

L'ambito di una variabile è il contesto nel quale viene definita; in genere esso abbraccia l'intero script PHP, tuttavia esistono alcune eccezioni, tra le quali troviamo le funzioni. Le variabili definite all'esterno delle funzioni non hanno alcun ambito all'interno della funzione, mentre le variabili definite all'interno della funzione non hanno ambito al di fuori di essa. Nel caso delle classi troviamo una situazione simile. L'ambito delle variabili all'interno delle funzioni è analizzato nel Capitolo 13, mentre le classi sono argomento del Capitolo 37.

Riepilogo

Questo capitolo ha introdotto il concetto di variabile, illustrando come possono essere dichiarate in PHP e come sia possibile assegnare loro i valori. Abbiamo anche descritto brevemente i diversi tipi di variabile riconosciuti da PHP. Nel prossimo capitolo verranno analizzati in modo più approfondito i tipi booleani, interi e in virgola mobile.

Capitolo 7

Tipi booleano, intero e in virgola mobile

Introduzione

In questo capitolo verranno esaminati in modo più dettagliato tre tipi di variabile: booleano, intero e in virgola mobile. Inizieremo tuttavia spiegando come PHP supporti le definizioni esplicite di tipi.

Tipi

PHP effettivamente non supporta le definizioni esplicite di tipi, a differenza di altri linguaggi di programmazione. In altre parole, al momento della creazione di una variabile non se ne specifica il tipo, che viene invece determinato dal contesto in cui essa viene utilizzata. Quindi, per esempio, se definite una variabile e le assegnate un intero, essa sarà di tipo intero; se in seguito le assegnate un valore in virgola mobile, la variabile diventerà di tipo in virgola mobile.

Tipi booleani

I booleani sono il tipo più semplice. Un valore booleano esprime un valore di verità che può essere TRUE o FALSE. L'esempio che segue illustra la dichiarazione di una variabile e l'assegnamento a essa del valore TRUE:

```
<?php

// Booleani, interi e float - Esempio 7-1
//--------------------

$weekday = TRUE;

echo $weekday;
?>
```

L'output di questo esempio è 1 e non il testo TRUE. Ciò avviene perché PHP utilizza il valore 1 per rappresentare TRUE e 0 per rappresentare FALSE; pertanto il contenuto di una variabile booleana sarà 1 o 0 una volta visualizzata.

Conversione a booleano

Come accennato nel capitolo precedente, per convertire in booleano possiamo utilizzare il cast `(bool)` o `(Boolean)`. Nella maggior parte dei casi non sarà necessario ricorrere a un cast esplicito, in quanto il valore sarà convertito automaticamente. Tuttavia, quando convertite a booleano da altri tipi di variabile, dovere sapere come verranno convertiti alcuni valori. La Tabella 7.1 offre un riepilogo di queste conversioni.

Tabella 7.1 Conversioni booleane

Convertito come	Conversione
FALSE	Un FALSE booleano.
FALSE	Il valore intero 0.
FALSE	Il valore in virgola mobile 0.0.
FALSE	Una stringa vuota e una stringa "0".
FALSE	Un array di zero elementi.
FALSE	Un oggetto con zero variabili membro.
FALSE	Il tipo speciale NULL.
FALSE	Tutti gli altri valori.

Interi

Un intero è un numero appartenente all'insieme {..., −2, −1, 0, 1, 2, ...}. La dimensione massima di un numero intero dipende dalla piattaforma, ma normalmente è nell'ordine dei 2 miliardi. I numeri interi possono essere specificati con la notazione decimale, come mostrato di seguito:

```
$var = 123;
$var2 = -123;
```

In questo esempio $var viene definita come intero positivo di valore 123, mentre $var2 è definita come intero negativo di valore −123. Come in alcuni altri linguaggi, PHP consente di definire gli interi in altre basi numeriche, in particolare ottale ed esadecimale:

```
$var = 0123;
$var2 = 0x1a;
```

In questo caso $var è definita come numero 123 ottale, equivalente a 83 in notazione decimale. I numeri ottali vengono definiti con uno 0 che precede il numero. A $var2 viene invece assegnato il numero 1a esadecimale, equivalente a 26 in decimale. I numeri esadecimali sono definiti anteponendo 0x al numero. Se non riuscite a cogliere perché 123 in ottale equivale a 83 in decimale o perché 1a esadecimale è equivalente a 26 in decimale, osservate la Figura 7.1.

	Centinaia	Decine	Unità	
Decimale		8	3	
=		8 x 10	+ 3	= 83

	64	8	Unità	
Ottali	1	2	3	
=	1 x 64	2 x 8	+ 3	= 83

	256	16	Unità	
Esadecimale		1	a	
=		1 x 16	+ 10	= 26

Figura 7.1
Rappresentazione di numeri decimali, ottali ed esadecimali.

Conversione a intero

È possibile convertire a intero con i cast `(int)` o `(Integer)`. La Tabella 7.2 elenca le specifiche conversioni di valore che avranno luogo.

Tabella 7.2 Conversioni a intero

Convertito come	Conversione
1	Un TRUE booleano.
0	Un FALSE booleano.
–	La conversione a un tipo in virgola mobile produrrà un numero arrotondato per difetto.
–	La conversione delle stringhe è descritta nel Capitolo 8.
–	La conversione da altri tipi non è definita nel linguaggio PHP.

Overflow del tipo intero

Se specificate un numero intero che supera i limiti del tipo intero, esso sarà interpretato automaticamente come valore in virgola mobile.

Funzione var_dump

Anche se le funzioni saranno analizzate in modo più approfondito più avanti, è utile introdurre a questo punto una funzione chiamata `var_dump()`, che visualizza il tipo di una variabile che le viene passata. Il formato della funzione è il seguente: La sintassi della funzione è la seguente.

```
Void var_dump(mixed variabile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
variabile	Mixed	La variabile che desideriamo esaminare.
Restituzione di var_dump()	Void	Non restituisce alcunché.

Esempio di funzione:

```
var_dump $(vab);
```

Possiamo utilizzare questa funzione per determinare il tipo delle nostre variabili. Considerate l'esempio seguente:

```
<?php
// Booleani, interi e float - Esempio 7-2
//---------------------

$smallInt = 1;

$bool = TRUE;
$largeInt = 12345678901234567890;

var_dump ($smallInt);
var_dump ($bool);
var_dump ($largeInt);
?>
```

L'output di questo script, illustrato nella Figura 7.2, mostra che `$smallInt` è di tipo intero, `$bool` è booleana, mentre `$largeInt` viene effettivamente memorizzata come valore in virgola mobile.

Tipi in virgola mobile

I numeri in virgola mobile (noti anche come `double`, `float` o `real`) possono essere specificati con il ricorso a una qualunque tra le forme sintattiche seguenti:

```
$var1 = 1.234;
$var2 = 1.2e3;
$var3 = 7E-10;
```

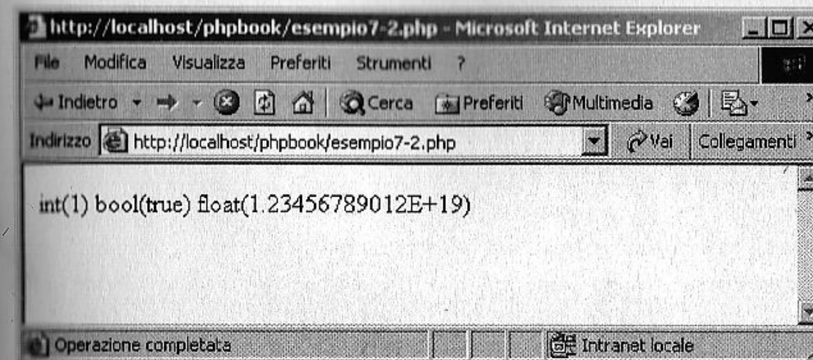



Figura 7.2
La funzione var_dump.

La dimensione di un numero in virgola mobile dipende dalla piattaforma, tuttavia potete considerare in generale un massimo di 1.8e308. I valori in virgola mobile presentano un problema di precisione. Considerate l'esempio seguente:

```
<?php

// Booleani, interi e float - Esempio 7-3
//--------------------

$one = 1;
$three = 3;
$third = $one/$three;

echo $third;
?>
```

In questo esempio il valore di `$third` è `0.333333333333`. I programmatori devono essere consapevoli di questa perdita di precisione quando si effettuano calcoli con i numeri in virgola mobile.

Conversione in virgola mobile

È possibile convertire a valore in virgola mobile con i cast `(float)`, `(double)` o `(real)`. Le conversioni di stringhe in valori in virgola mobile verranno descritte nel prossimo capitolo, nel paragrafo dedicato alla conversione delle stringhe in numeri.

Riepilogo

In questo capitolo sono stati analizzati i tipi booleano, intero e in virgola mobile, illustrando come possono essere definiti e come sia possibile convertire un tipo in un altro. Nel prossimo capitolo verranno analizzati in modo più approfondito i tipi di variabile stringa.

Capitolo 8

Stringhe

Introduzione

In questo capitolo verrà esaminato il tipo di variabile stringa. Le stringhe sono sequenze di caratteri alfanumerici e sono già state utilizzate in uno o due esempi di script PHP, in particolare come parte dell'istruzione `echo`. Inizieremo esaminando come vengono definite le stringhe in PHP, per poi analizzare i diversi tipi di stringa a disposizione. Dovremo poi capire come unire le stringhe, accedere ai singoli caratteri di una stringa e convertire le stringhe in numeri.

Tipi stringa

In PHP la specificazione delle stringhe può avvenire in due modi: è infatti possibile utilizzare i caratteri "'" o """ per indicare l'inizio e la fine della stringa. Per esempio:

```
$myString = 'Prima stringa';
$anotherString = "Un'altra stringa";
```

In questo esempio `$myString` viene definita come stringa contenente i caratteri `'Prima stringa'`, mentre `$anotherString` viene definita come stringa contenente i caratteri `"Un'altra stringa"`. Esiste allora qualche differenza tra le due forme? La risposta può essere sì o no in relazione al contesto in cui viene impiegata la stringa. Per esempio:

```
<?php
// Stringhe - Esempio 8-1
//---------------------

$myString = 'Prima stringa';
$anotherString = "Un'altra stringa";

echo $myString;
echo "<br>";
echo $anotherString;
?>
```

L'output dell'esempio precedente è il seguente:

```
Prima stringa
Un'altra stringa
```

Sembrerebbe quindi che le stringhe con virgolette singole e quelle con virgolette doppie siano sostanzialmente identiche. Questo tuttavia non è del tutto vero, come si vedrà quando verrà analizzato in modo più approfondito il supporto che queste diverse stringhe hanno per le sequenze di escape per i caratteri.

Stringhe con virgolette singole

Le stringhe con virgolette singole possono contenere solo le sequenze di escape elencate nella Tabella 8.1.

Tabella 8.1 Sequenze di escape per i caratteri nelle stringhe con virgolette singole

Sequenza	Descrizione
\\	Carattere barra contraria.
\'	Carattere virgoletta singola.

Queste sequenze di escape consentono di includere i caratteri "\" e "'" nelle stringhe.
Considerate l'esempio seguente:

```
<?php

// Stringhe - Esempio 8-2
//---------------------

$myString = 'Questa è una barra rovesciata: \\';
$anotherString = ' e questa è una virgoletta singola: \'';

echo $myString;
echo $anotherString;

?>
```

Ecco cosa visualizza lo script precedente:

```
Questa è una barra rovesciata: \ e questa è una virgoletta singola: '
```

I motivi per cui è necessario ricorrere alle sequenze di escape per questi caratteri sono due.

1. Se desideriamo visualizzare una virgoletta singola come parte della stringa, dobbiamo segnalare al parser PHP che si tratta di un carattere da visualizzare e non di un virgoletta che indica la fine della stringa.
2. Utilizziamo il carattere barra rovesciata per indicare una sequenza di escape, pertanto dobbiamo ricorrere a una sequenza di escape quando desideriamo visualizzare proprio questo carattere.

Pertanto l'istruzione seguente produrrebbe un errore se venisse inserita in uno script:

```
$var3 = 'Questa virgoletta ' invece produrrebbe un errore';
```

Per avere una visualizzazione corretta, la riga dovrebbe essere scritta nel modo seguente:

```
$var3 = 'Questa virgoletta \' invece produrrebbe un errore';
```

Stringhe con virgolette doppie

Con le stringhe a virgolette doppie, invece, le cose si complicano leggermente, a cominciare dal numero di sequenze di escape per i caratteri che è possibile utilizzare, che è molto più elevato. Osservate la Tabella 8.2.

Tabella 8.2 Sequenze di escape per i caratteri nelle stringhe con virgolette doppie

Sequenza	Descrizione
\n	Ritorno carrello e avanzamento riga.
\r	Ritorno carrello.
\t	Carattere di tabulazione.
\\	Barra rovesciata.
\$	Simbolo del dollaro.
\"	Virgolette doppie.
\[0-7]	Un carattere specificato in notazione ottale con 1-3 cifre. Il primo numero è uno 0.
\x[0-9A-Fa-f]	Un carattere specificato in notazione esadecimale con 1 o 2 cifre. Deve avere un carattere x prima del numero.

La sequenza \n costringe il cursore a saltare all'inizio di una nuova riga; la sequenza \r costringe il cursore a saltare all'inizio della riga corrente; la sequenza \t inserisce un carattere di tabulazione. Osservando gli esempi di questo libro noterete che queste sequenze di escape non sono molto utilizzate.
La sequenza \\ inserisce un carattere barra contraria; la sequenza \$ inserisce il carattere dollaro; la sequenza \" inserisce un carattere di virgolette doppie; le due sequenze di escape finali consentono di specificare i singoli caratteri sotto forma di numeri ottali o esadecimali, in conformità con lo standard ASCII (*American Standard Code for Information Interchange*), un codice che assegna un valore numerico a ogni lettera, numero e carattere presente sulla tastiera (e ad alcuni caratteri che non vi compaiono).
Pertanto la stringa seguente:

```
$var = "Ciao \x4C\x69\x7A";
```

produrrebbe la stringa:

```
Ciao Liz
```

Perché? 4C è un numero esadecimale equivalente a 76 decimale; il carattere 76 nella tabella ASCII corrisponde al carattere 'L'. 69 equivale a 105 in decimale e corrisponde al carattere 'i'. Infine 7A equivale a 122 in decimale e corrisponde al carattere 'z'.

Stringhe heredoc

Un altro modo per definire le stringhe implica il ricorso alla sintassi heredoc, che utilizza tre caratteri di minore (<<<) seguiti da un identificatore. È necessario utilizzare un identificatore di chiusura per indicare la fine della stringa. Tale identificatore deve utilizzare gli stessi caratteri dell'identificatore di inizio, deve iniziare alla prima colonna della riga e dev'essere seguito immediatamente da un punto e virgola. Per esempio:

```
<?php

// Stringhe - Esempio 8-3
//---------------------

$myString = <<<MST
Questo è un esempio di stringa
che occupa più righe
e utilizza la sintassi heredoc
MST;

echo $myString;
?>
```

La stringa heredoc si comporta esattamente come quella con virgolette doppie.

Variabili nelle stringhe

Le stringhe con virgolette doppie consentono di utilizzare un numero maggiore di sequenze di escape rispetto a quelle con virgolette singole. Perché allora non utilizzarle sempre? In effetti c'è un'altra differenza tra le due stringhe: qualunque variabile inclusa in una stringa con virgolette doppie verrà espansa. Che cosa significa? Considerate l'esempio che segue:

```
$name = "Simone";
$message = "Ciao, $name";
$message = 'Ciao, $name';
```

Nel frammento di script precedente, la prima riga definisce una stringa con valore `"Simone"`; la seconda riga definisce una stringa con valore `"Ciao, Simone"`, mentre la terza definisce una stringa con valore `'Ciao, $name'`. Questo dipende dalla differenza tra stringhe a virgolette singole e a virgolette doppie. Con una stringa a virgolette singole la variabile (in questo caso `$name`) non viene riconosciuta come tale e viene visualizzato il testo `'$name'`; nel caso invece di una stringa che utilizza le virgolette doppie, la variabile viene riconosciuta e viene sostituita con il valore corrente.
È possibile assegnare a una stringa il valore di un'altra stringa utilizzando il carattere o operatore uguale (=). Per esempio:

```
$name = "Simone";
$anotherName = $name;
```


Qui il valore della variabile `$name` viene assegnato alla variabile `$anotherName` (ossia copiato in essa). Lo script seguente illustra l'uso delle variabili nelle stringhe:

```
<?php
// Stringhe - Esempio 8-4
//---------------------

$firstname = 'Liz';
$surname = 'Hall';
$message = "Ciao, $firstname";
$message2 = "Il tuo nome completo è $firstname $surname";

echo $message;
echo "<br />";
echo $message2;
echo "<br />";
echo "Arrivederci, Sig.ra $surname";

?>
```

L'esempio precedente produce l'output illustrato nella Figura 8.1.

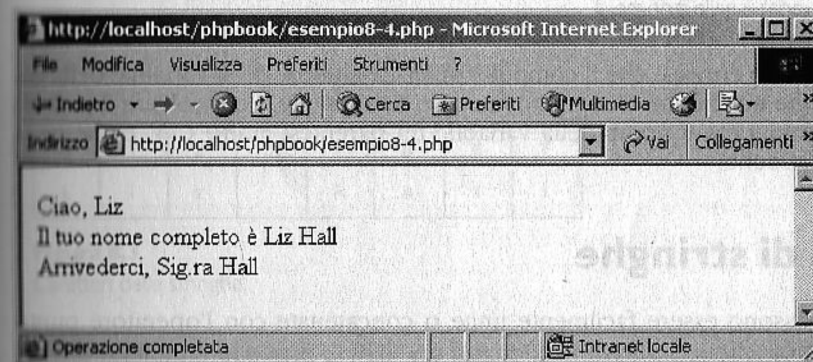


Figura 8.1
Variabili nelle stringhe.

Dovete sapere però che le stringhe non funzionano sempre esattamente nel modo desiderato. Considerate lo script seguente:

```
<?php
// Stringhe - Esempio 8-5
//---------------------

$currency = "dollar";

echo "The currency is the $currency<br />";

echo "I would like some $currencys <br />";

echo "I would like some ${currency}s <br />"
?>
```

Lo script precedente genera un errore, come mostrato nella Figura 8.2.

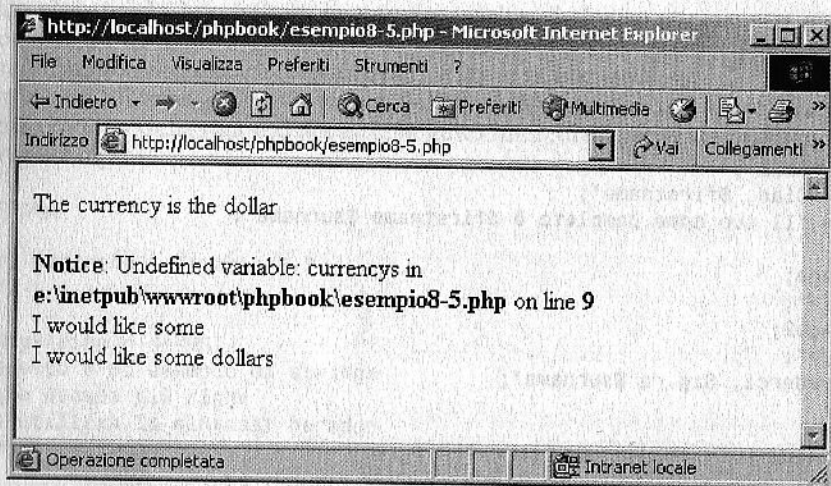


Figura 8.2
Problemi con le variabili nelle stringhe.

Il motivo di questo errore è che il computer analizza il nome della variabile della seconda istruzione `echo` come `$currencys` e non `$currency`. Per risolvere il problema potete racchiudere il nome della variabile tra parentesi graffe `{ }`, come nella terza istruzione `echo`.

Unione di stringhe

Le stringhe possono essere facilmente unite o concatenate con l'operatore punto ".". Per esempio:

```
$title = "Sig.ra ";
$firstName = "Liz ";
$surName = "Hall";
$fullName = $firstName . $surName;
$titleName = $title . $firstName . $surName;
```

Il frammento di codice precedente crea una stringa `$fullName` con valore "Liz Hall". Osservate che lo spazio tra il nome e il cognome non verrebbe incluso se non ci fosse un carattere di spazio alla fine della stringa "Liz ". `$titleName` viene creata con il valore "Sig.ra Liz Hall".
Possiamo anche utilizzare questo operatore nella nostra istruzione `echo` per visualizzare le variabili in modo più chiaro. Considerate l'esempio seguente:

```
<?php

// Stringhe - Esempio 8-6
//---------------------

$tax = 17;
```



```
$net = 200;
$copyOfNet = &$net;
$anotherCopy = $net;

$net = 100;

echo "$copyOfNet $anotherCopy";
echo "<br>";
echo $copyOfNet . " " . $anotherCopy;
?>
```

L'esempio precedente illustra come sia possibile impiegare l'operatore punto per unire stringhe in un'istruzione `echo`.

Accesso ai caratteri delle stringhe

L'accesso ai caratteri di una stringa può avvenire specificando tra parentesi graffe dopo la stringa l'offset rispetto a zero del carattere desiderato. Per esempio:

```
$name = "Liz Hall";
$second = $name{1};
```

La Figura 8.3 illustra come viene memorizzata la stringa e come avviene il riferimento a essa. La variabile `$second` ha valore "i".

0	1	2	3	4	5	6	7
L	i	z		H	a	l	l

Figura 8.3
Caratteri delle stringhe.

Nelle versioni precedenti di PHP era possibile utilizzare i caratteri `[ ]` invece di `{ }`, ma oggi essi non sono più validi. Lo script seguente illustra l'uso della concatenazione delle stringhe e il riferimento ai caratteri:

```
<?php

// Stringhe - Esempio 8-7
//---------------------

$firstName = 'Liz ';
$surName = 'Hall';
$fullName = $firstName . $surName;

$first = $fullName{0};
$second = $fullName{1};
$third = $fullName{2};
$fourth = $fullName{3};
$fifth = $fullName{4};
$sixth = $fullName{5};
```

```
$seventh = $fullName{6};
$eighth = $fullName{7};

echo "Il nome completo è : " . $fullName . "<p>";
echo "Il primo carattere in $fullName è $first<br>";
echo "Il secondo carattere in $fullName è $second<br>";
echo "Il terzo carattere in $fullName è $third<br>";
echo "Il quarto carattere in $fullName è $fourth<br>";
echo "Il quinto carattere in $fullName è $fifth<br>";
echo "Il sesto carattere in $fullName è $sixth<br>";
echo "Il settimo carattere in $fullName è $seventh<br>";
echo "L'ottavo carattere in $fullName è $eighth<br>";
?>
```

L'output prodotto dallo script precedente è illustrato nella Figura 8.4.

Conversione di stringhe in numeri

In alcuni casi le stringhe possono essere valutate come valori numerici. Il valore e il tipo numerico sono determinati nel modo indicato di seguito.

1. La stringa sarà valutata come valore in virgola mobile se contiene uno qualunque dei caratteri seguenti, ".", "e" o "E", altrimenti verrà valutata come intero.
2. Il valore numerico sarà determinato dalla parte iniziale della stringa. Se si tratta di un valore numerico valido verrà utilizzato, altrimenti la stringa sarà valutata `0`.

Lo script che segue illustra questa conversione:

```
<?php
// Stringhe - Esempio 8-8
//-------------------

$num = 23;

$string = "45";
$add = $num + $string;
$males = "3.45 maschi";
$females = 2 + $males;

echo "23 + '45' è uguale a $add<br />";
echo "2 + '3.45' è uguale a $females<br />";
?>
```

L'output prodotto da questo script è il seguente:

```
23 + '45' è uguale a 68
2 + '3.45' è uguale a 5.45
```

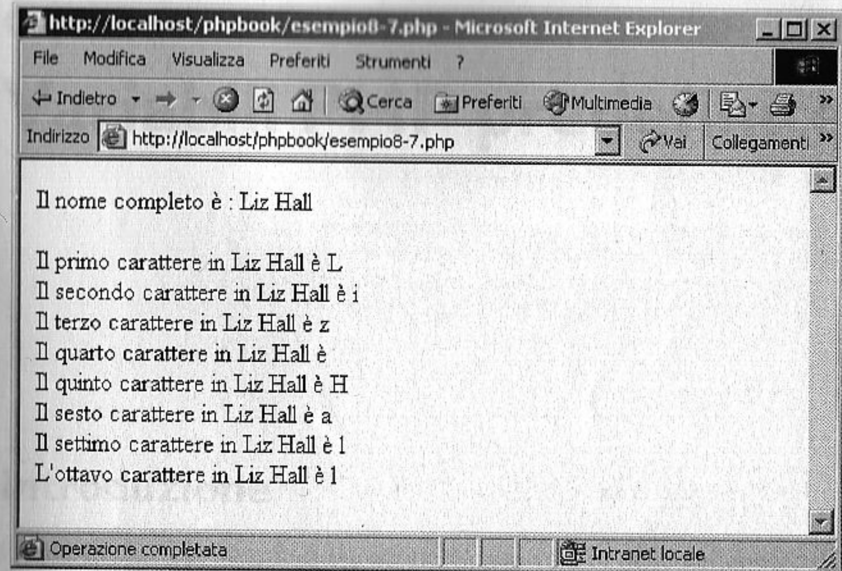



Figura 8.4
Riferimento a caratteri singoli in una stringa.

Riepilogo

Questo capitolo ha introdotto il tipo stringa, esplorando i diversi modi in cui è possibile definire una stringa in PHP e come questi diversi tipi di stringa influiscano sulle modalità di impiego delle stringhe stesse. Abbiamo poi analizzato come unire più stringhe, accedere ai singoli caratteri e convertire le stringhe in numeri. Nel prossimo capitolo verrà introdotto il concetto di variabili dall'esterno di PHP.

Capitolo 9

Variabili PHP predefinite

Introduzione

In questo capitolo verranno introdotte alcune variabili che vengono prodotte al di fuori di PHP e che sono note come variabili predefinite. Si vedrà come il server Web, l'ambiente del sistema e i moduli HTML generano variabili cui è possibile (e in molti casi necessario) accedere. Le variabili predefinite sono disponibili per qualunque script, ma poiché molte di esse dipendono dalla piattaforma e dal server Web in uso, è certamente difficile documentarle tutte. Inizieremo illustrando un cambiamento di fondo avvenuto in PHP a partire dalla versione 4.2 e che influisce notevolmente sul modo in cui avviene l'accesso a queste variabili predefinite.

Direttiva register_globals

A partire dal rilascio della versione 4.2.0, PHP viene distribuito con la direttiva `register_globals` (situata nel file php.ini) impostata su off. Ciò significa che le variabili relative a moduli, server, cookie e così via non vengono più registrate come variabili globali. Per PHP si tratta di un cambiamento di grande importanza che accresce notevolmente la sicurezza degli script. Ciò significa, tuttavia, che gli script PHP realizzati prima della versione 4.2.0 non funzioneranno correttamente se non seguono la nuova forma di accesso alle variabili predefinite (come avviene negli esempi di questo libro) oppure se non si imposta su on la direttiva `register_globals` nel file php.ini (soluzione sconsigliata).

Funzione phpinfo()

PHP include una funzione molto utile chiamata `phpinfo()`, che visualizza una grande quantità di informazioni sullo stato corrente di PHP, tra cui le opzioni di compilazione attivate, le estensioni attivate e le variabili predefinite disponibili. Nel nostro caso è particolarmente utile, in quanto verranno visualizzate tutte le variabili predefinite definite dal sistema.
La sintassi della funzione è la seguente.

```
int phpinfo(int opzione);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
opzione	Int	Opzioni di configurazione, valori (1, 2, 4, 8, 16, 32, 64 e –1): 1 informazioni generali; 2 crediti di PHP; 4 configurazione (direttive PHP impostate nel file php.ini); 8 moduli caricati; 16 variabili d'ambiente; 32 variabili predefinite; 64 informazioni sulla licenza; –1 tutte le precedenti; è l'impostazione predefinita se non viene incluso alcun valore.
Restituzione di `phpinfo`	Int	Restituisce `TRUE` se riesce o `FALSE` se si verifica un errore.

Esempio di funzione:

```
phpinfo(32);
```

Lo script seguente illustra l'utilizzo di questa funzione:

```
<?php

// Variabili predefinite - Esempio 9-1
//-----------------------

phpinfo(32);

?>
```

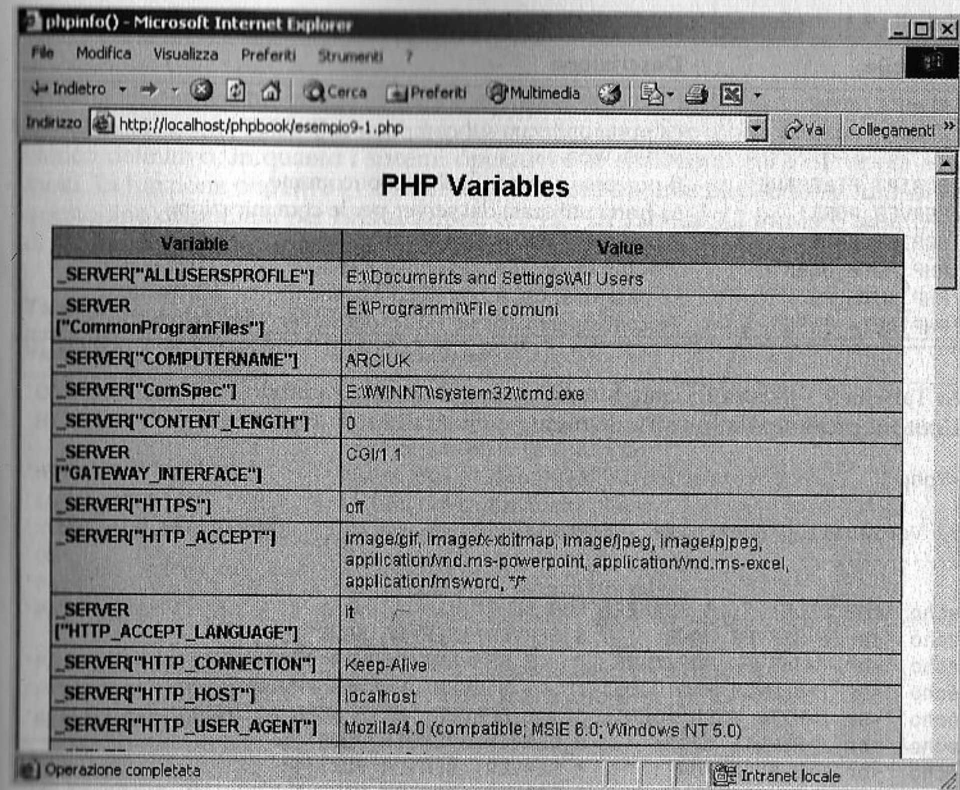
L'output di questo script è illustrato nella Figura 9.1.

Variabili del server

Le variabili del server sono quelle impostate dal server Web. Poiché il server Web di ciascun produttore è un prodotto sostanzialmente diverso dagli altri, non c'è alcuna garanzia che tutti i server Web forniscano queste variabili, anche se la maggior parte di essi ne fornirà almeno qualcuna. La Tabella 9.1 elenca le variabili del server più comuni e una descrizione del loro contenuto.

Le variabili del server (come tutte le variabili predefinite descritte in questo capitolo) sono memorizzate in un array predefinito. Nel Capitolo 14 imparerete a creare i vostri array, ma per il momento dovete semplicemente capire come si accede alle variabili predefinite memorizzate in un array. Per accedere alle variabili del server viene utilizzato il nome di array `$_SERVER` seguito da una coppia di parentesi quadre che racchiude tra virgolette singole il nome della variabile predefinita. Per esempio, il frammento seguente memorizza il nome dello script nella variabile `$script`.

```
$script = $_SERVER['PHP_SELF'];
```

Variable	Value
_SERVER["ALLUSERSPROFILE"]	E:\Documents and Settings\All Users
_SERVER ["CommonProgramFiles"]	E:\Programmi\File comuni
_SERVER["COMPUTERNAME"]	ARCIUK
_SERVER["ComSpec"]	E:\WINNT\system32\cmd.exe
_SERVER["CONTENT_LENGTH"]	0
_SERVER ["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["HTTPS"]	off
_SERVER["HTTP_ACCEPT"]	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
_SERVER ["HTTP_ACCEPT_LANGUAGE"]	it
_SERVER["HTTP_CONNECTION"]	Keep-Alive
_SERVER["HTTP_HOST"]	localhost
_SERVER["HTTP_USER_AGENT"]	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

Figura 9.1
Output di phpinfo().

Tabella 9.1 Variabili del server

Variabile	Descrizione
`'PHP_SELF'`	Il nome di file dello script corrente relativo alla directory root dei documenti. Verrà utilizzata in molti script proposti più avanti nel prosieguo del libro, laddove verranno introdotti i moduli e sarà necessario fare riferimento allo script medesimo.
`'GATEWAY_INTERFACE'`	La versione dell'interfaccia CGI in uso.
`'SERVER_NAME'`	Il nome del server host.
`'SERVER_SOFTWARES'`	Stringa di identificazione del server che viene passata nelle intestazioni HTML quando risponde alle richieste.
`'SERVER_PROTOCOL'`	Nome e versione del protocollo di informazioni con il quale la pagina è stata richiesta.
`'REQUEST_METHOD'`	Il metodo di richiesta utilizzato per accedere alla pagina, ossia `'POST'`.
`'QUERY_STRING'`	La stringa di query (se presente) utilizzata nell'accesso alla pagina.
`'DOCUMENT_ROOT'`	Il file root dei documenti, definito nel file di configurazione del server.
`'HTTP_REFERER'`	L'indirizzo della pagina Web che ha portato l'agente utente (browser Web) alla pagina corrente.
`'HTTP_USER_AGENT'`	L'intestazione dell'agente utente che accede alla pagina.

(*Segue*)

Tabella 9.1 *(Continua)* Variabili del server

Variabile	Descrizione
'REMOTE_ADDR'	L'indirizzo IP dal quale l'utente visualizza la pagina.
'REMOTE_PORT'	La porta utilizzata sul computer dell'utente per comunicare con il server Web.
'SCRIPT_FILENAME'	Il percorso assoluto dello script corrente.
'SERVER_PORT'	La porta utilizzata dal server per le comunicazioni.
'SCRIPT_NAME'	Il percorso dello script corrente.
'PHP_AUTH_USER'	Il nome utente del server Web Apache fornito dall'utente.
'PHP_AUTH_PW'	La password del server Web Apache fornita dall'utente.
'PHP_AUTH_TYPE'	Tipo di autenticazione HTTP del server Web Apache.

La Tabella 9.1 illustra il contenuto di alcune di queste variabili.
L'output prodotto dallo script seguente è illustrato nella Figura 9.2.

```
<?php

// Variabili predefinite - Esempio 9-2
//-------------------

echo "PHP_SELF: " .$_SERVER['PHP_SELF'];
echo "<br />SERVER_NAME: " . $_SERVER['SERVER_NAME'];
echo "<br />SERVER_SOFTWARE: " . $_SERVER['SERVER_SOFTWARE'];
echo "<br />SERVER_PROTOCOL: " . $_SERVER['SERVER_PROTOCOL'];
echo "<br />HTTP_USER_AGENT: " . $_SERVER['HTTP_USER_AGENT'];
echo "<br />REMOTE_ADDR: " . $_SERVER['REMOTE_ADDR'];
echo "<br />SERVER_PORT: " . $_SERVER['SERVER_PORT'];
echo "<br />SCRIPT_NAME: " . $_SERVER['SCRIPT_NAME'];

?>
```

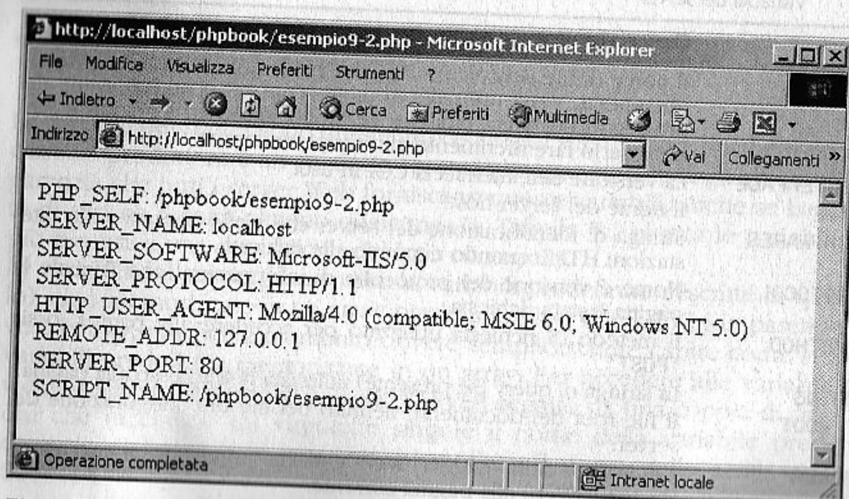


Figura 9.2
Le variabili predefinite del server.

Variabili d'ambiente

Le variabili d'ambiente vengono messe a disposizione di PHP dal sistema operativo sul quale si esegue PHP. Questo è il motivo per cui sarebbe molto difficile offrire un elenco definitivo, in quanto i sistemi operativi e le shell sotto cui PHP opera sono molti. La funzione `phpinfo()` consente di ottenere un utile elenco delle funzioni di questo tipo disponibili per l'installazione di PHP. La Tabella 9.2 fornisce descrizioni di alcune di queste variabili nell'ambito di un'installazione di PHP in Windows XP.

Tabella 9.2 Variabili d'ambiente

Variabile	Descrizione
'COMPUTERNAME'	Il nome del computer.
'HTTP_HOST'	L'indirizzo del server Web (localhost è quello più comune per gli ambienti di sviluppo).
'HTTP_USER_AGENT'	La stringa di identificazione del browser Web.
'LOCAL_ADDR'	L'URL del computer locale.
'NUMBER_OF_PROCESSORS'	Il numero di processori.
'OS'	Il sistema operativo.
'Path'	Il percorso del sistema operativo.
'PATH_INFO'	Il percorso dello script corrente a partire dalla directory root degli script.
'PATH_TRANSLATED'	Il percorso completo dello script corrente.
'PROCESSOR_IDENTIFIER'	La stringa di identificazione del processore.
'SERVER_SOFTWARE'	La stringa di identificazione del server Web.

Per accedere alle variabili del server viene utilizzato il nome di array `$_ENV` seguito da una coppia di parentesi quadre che racchiude tra virgolette singole il nome della variabile predefinita. Per esempio:

```
$os = $_ENV['S'];
```

Questa istruzione memorizza il sistema operativo nella variabile `$os`. Lo script seguente illustra il contenuto di alcune di queste variabili:

```
<?php

// Variabili predefinite - Esempio 9-3
//-------------------

echo "COMPUTERNAME: " . $_ENV['COMPUTERNAME'];
echo "<br />HTTP_HOST: " . $_ENV['HTTP_HOST'];
echo "<br />HTTP_USER_AGENT: " . $_ENV['HTTP_USER_AGENT'];
echo "<br />LOCAL_ADDR: " . $_ENV['LOCAL_ADDR'];
echo "<br />NUMBER_OF_PROCESSORS: " . $_ENV['NUMBER_OF_PROCESSORS'];
echo "<br />OS: " . $_ENV['OS'];
echo "<br />Path: " . $_ENV['Path'];
echo "<br />PATH_INFO: " . $_ENV['PATH_INFO'];
echo "<br />PATH_TRANSLATED: " . $_ENV['PATH_TRANSLATED'];
echo "<br />PROCESSOR_IDENTIFIER: " . $_ENV['PROCESSOR_IDENTIFIER'];
echo "<br />SERVER_SOFTWARE: " . $_ENV['SERVER_SOFTWARE'];

?>
```

L'output prodotto dallo script precedente è illustrato nella Figura 9.3.

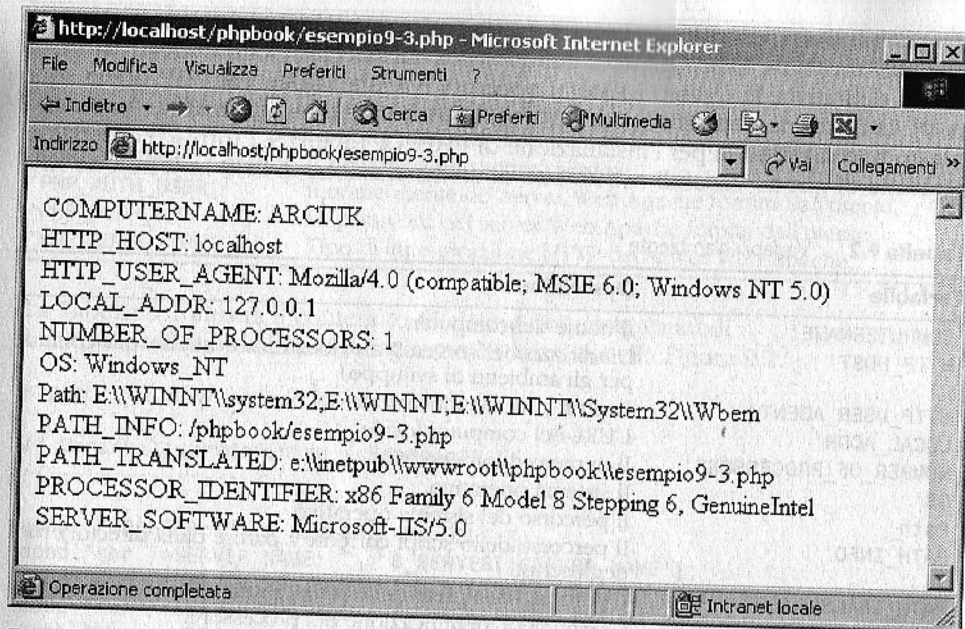


Figura 9.3
Variabili di ambiente predefinite.

Variabili dei moduli

Nel Capitolo 18 verrà introdotto il concetto di interazione con l'utente attraverso i moduli. Vedrete come i moduli consentano all'utente di inserire dati che a loro volta vengono memorizzati in variabili, cui è possibile accedere tramite lo script PHP. I dati dei moduli che vengono passati allo script tramite il metodo `POST` vengono memorizzati in un array chiamato `$_POST` e l'accesso a essi può avvenire come per le variabili del server. Per esempio:

```
$formField = $_POST ['formField'];
```

Oltre al metodo `POST`, il passaggio dei dati può avvenire anche con il metodo `GET`. I dati di un modulo passati tramite il metodo `GET` possono essere utilizzati accedendo all'array `$_GET`. Per esempio:

```
$dataField = $_GET['dataField'];
```

Gli esempi del libro utilizzeranno soltanto il metodo `POST`.

Variabili di sessione

Nel Capitolo 23 verranno introdotte le sessioni e sarà fornita una spiegazione di come possano essere impiegate per fornire una pagina Web personalizzata a livello individuale per ciascun utente Web. Le variabili memorizzate per ciascun utente sono detenute in file di sessione e l'accesso a esse è possibile attraverso l'array `$_SESSION`. Per esempio:

```
$sessData = $_SESSION['sessData'];
```

Variabili dei cookie

Nel Capitolo 22 si parlerà dei cookie e si vedrà come possano essere utilizzati per memorizzare i dati di un utente specifico a livello locale sul computer del client. L'accesso alle variabili dei cookie può avvenire attraverso l'array `$_COOKIE`. Per esempio:

```
$cookieData = $_COOKIE['cookieData'];
```

Costrutto isset()

Prima di concludere il capitolo vale la pena di menzionare il costrutto `isset()`, che può essere utilizzato per controllare se una variabile è stata o meno impostata. La sintassi della funzione è la seguente.

```
bool isset(mixed variabile)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
variabile	Mixed	La variabile da controllare.
Restituzione di `isset()`	bool	`TRUE` se la variabile è impostata, `FALSE` se non lo è.

Esempio di funzione:

```
isset ($_ENV['OS']);
```

Questo costrutto verrà utilizzato insieme al costrutto `if` nei capitoli successivi per determinare se alcune variabili predefinite sono state o meno impostate, in modo da evitare la generazione di messaggi di errore nel caso in cui si cercasse di accedere a una variabile che non è impostata. Per esempio:

```
<?php

// Variabili predefinite - Esempio 9-4

echo "<br />QUERY_STRING: " . $_SERVER['QUERY_STRING'];

?>
```

Lo script precedente dovrebbe restituire un avviso come il seguente:

```
Notice: Undefined index: QUERY_STRING in
c:\inetpub\wwwroot\phpbook\esempio9-4.php on line 6
QUERY_STRING:
```

Più avanti vedremo come sia possibile utilizzare `isset()` per evitare questo messaggio.

Riepilogo

Questo capitolo ha presentato i diversi tipi di variabili predefinite disponibili in PHP. Non sono state analizzate tutte in modo approfondito, in quanto verranno introdotte al momento opportuno nei capitoli successivi. Tuttavia è importante osservare che abbiamo utilizzato solo la forma sicura delle variabili predefinite introdotta nella versione 4.2.0 di PHP, dove le variabili globali non sono registrate per un accesso globale. Argomento del prossimo capitolo saranno espressioni, operandi e operatori.

Capitolo 10

Espressioni, operandi e operatori

Introduzione

Nei capitoli precedenti sono stati esaminati i vari tipi di variabili; ora, invece, l'attenzione verrà rivolta ai modi per manipolare le informazioni contenute in esse con l'introduzione dei concetti di espressioni, operandi e operatori. Le espressioni vengono utilizzate per calcolare un risultato che implica diversi operandi (variabili o costanti); gli operatori consentono di manipolare le variabili e le costanti nelle espressioni; gli operandi sono le costanti o le variabili su cui agiscono gli operatori.

Espressioni

In programmazione, un'espressione è un elemento che *esprime* un valore. Quest'ultimo può essere un semplice valore, ossia un singolo operando che esprime un singolo valore:

```
5
"Testo"
$place
```

oppure un insieme di operandi che vengono combinati tramite operatori:

```
2 + 7 + 15,
"il significato della" . "vita"
"Benvenuti a " . $place
```

Gli operandi di un'espressione possono essere variabili, costanti o altre sottoespressioni. La Figura 10.1 illustra la differenza tra operandi, operatori ed espressioni in due semplici esempi.
Ecco una semplice espressione:

```
$var = 5;
```

Qui alla variabile `$var` viene assegnato il valore costante 5. In questo esempio 5 è l'espressione.

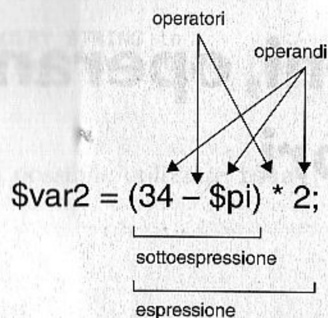


Figura 10.1
Operandi, operatori ed espressioni.

Segue un altro esempio:

```
$var = $var2;
```

In questo caso a `$var` viene assegnato il valore di `$var2`. In questo esempio `$var2` è l'espressione.
Considerate ora l'espressione seguente:

```
$var = 5 + 3 + 2;
```

L'espressione `5 + 3 + 2` è costituita da due operatori e tre operandi e non presenta particolari difficoltà di comprensione, in quanto le variabili 5, 3 e 2 vengono sommate e il risultato 10 viene memorizzato in `$var`. Vediamo tuttavia cosa accade se utilizziamo un operatore – (meno) invece dell'operatore +:

```
$var = 5 - 3 - 2;
```

Questa espressione significa forse sottrarre 3 da 5, che dà 2, e poi sottrarre 2 da questo, ottenendo 0? Oppure significa che 2 verrà sottratto da 3, ottenendo 1, che poi verrà sottratto da 5, con risultato finale 4? Fortunatamente è possibile utilizzare le parentesi per chiarire come dev'essere intesa l'espressione:

```
$var = (5 - 3) - 2;
$var = 5 - (3 - 2);
```

Più avanti nel capitolo verrà esaminato l'ordine in cui vengono valutate le espressioni.

Operatori

Gli operatori consentono di manipolare le variabili e le costanti, o di "operare" su di esse, e benché probabilmente li conosciate già, potreste non averli mai sentiti chiamare con il termine di operatori. Gli operatori più noti saranno quelli per le operazioni matematiche di addizione (+), sottrazione (–), divisione (/) e moltiplicazione

(*). Gli operatori vengono descritti come unari, binari o ternari secondo la loro capacità di accettare uno, due o tre operandi (argomenti). Gli operatori più, meno, moltiplicazione e divisione sono tutti operatori binari.

Operandi

Un operando non è altro che un elemento sul quale un operatore lavora. In un'espressione come `34 - $pi`, un operando è una costante (34), mentre l'altro è una variabile (`$pi`).

Operatori aritmetici

PHP supporta cinque diversi operatori aritmetici, elencati nella Tabella 10.1.

Tabella 10.1 Operatori aritmetici

Nome	Operatore	Esempio	Descrizione
Addizione	+	`$a + $b`	Somma `$a` e `$b`.
Sottrazione	–	`$a - $b`	Sottrae `$b` da `$a`.
Moltiplicazione	*	`$a * $b`	Moltiplica `$a` e `$b`.
Divisione	/	`$a/$b`	Divide `$a` per `$b`.
Modulo	%	`$a % $b`	Resto della divisione di `$a` per `$b`.

Lo script che segue offre un esempio di utilizzo degli operatori aritmetici:

```
<?php
// Espressioni - Esempio 10-1
//------------------------

$a = 5;
$b = 3;

echo 'a + b = ' . ($a + $b) . '<br>';
echo 'a - b = ' . ($a - $b) . '<br>';
echo 'a * b = ' . ($a * $b) . '<br>';
echo 'a/b = ' . ($a/$b) . '<br>';
echo 'a % b = ' . ($a % $b) . '<br>';

?>
```

La Figura 10.2 illustra l'output prodotto da questo script.
Osservate che nell'esempio l'operatore `%` restituisce il resto della divisione di 5 per 3, ossia 2.

Operatori di assegnamento

La descrizione delle espressioni fornita in precedenza è oltremodo semplificata. Il segno di uguale (=) in PHP è un operatore di per sé, l'operatore di assegnamento,

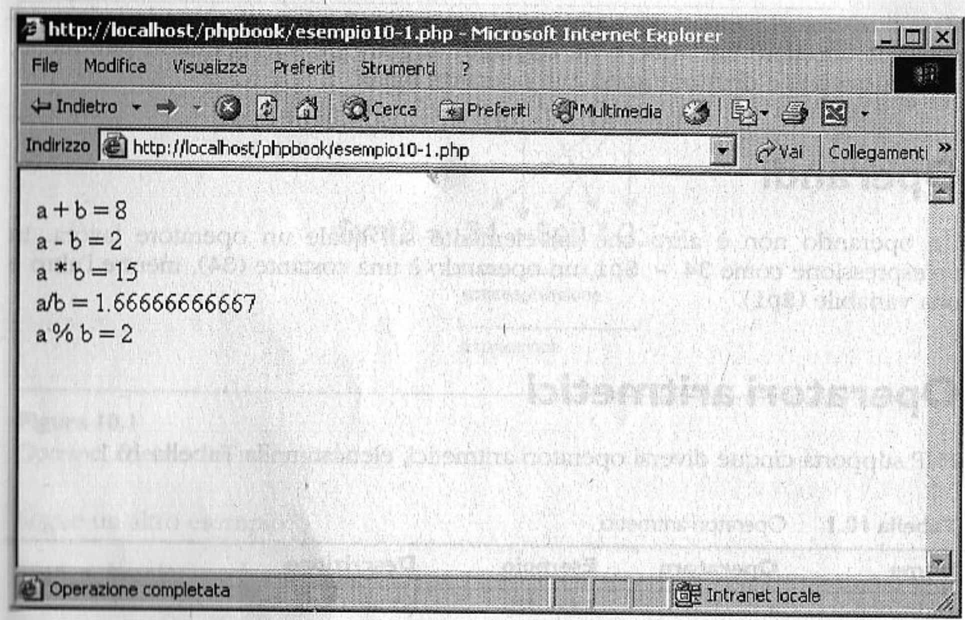


Figura 10.2
Utilizzo degli operatori aritmetici.

che agisce impostando l'operando alla sua sinistra al valore dell'operando posto sulla destra. Ciò significa che `$a = 5 + 3`, per esempio, costituisce un'espressione nel complesso (non solo la parte alla destra di "=", quindi). La Figura 10.3 illustra quanto appena affermato.

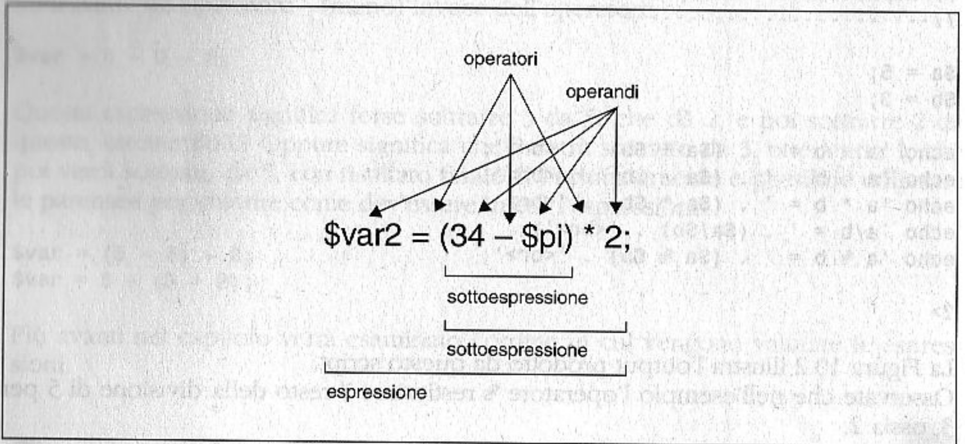


Figura 10.3
Un'espressione completa.

Non dovete considerare "=" come un carattere di "uguale a", poiché in realtà dovrebbe essere letto come carattere "assegna a". L'uso più semplice dell'operatore di assegnamento è il seguente:

```
$var = 56;
```

Questa istruzione significa "il valore 56 viene assegnato alla variabile `$var`". Ecco ora un esempio un po' più complesso:

```
$var = $a = $b;
```

Questa riga significa che il valore contenuto in `$b` viene assegnato a `$a`, che a sua volta viene assegnata a `$var`. PHP supporta anche i seguenti "operatori combinati":

```
$a = 3;
$a += 4;
$b = "Salve ";
$b .= "a tutti!";
```

La prima riga assegna il valore `3` alla variabile `$a`.

```
$a = 3;
```

La seconda riga assegna il valore `7` a `$a` proprio come se fosse stato scritto:

```
$a = $a + 4;
```

La terza e la quarta riga producono il valore `"Salve a tutti!"` assegnato `$b`, che equivale a scrivere:

```
$b = "Salve ";
$b = $b . "a tutti!";
```

La forma dell'operatore di assegnamento non ha importanza; scegliete quello che trovate più comodo.

Operatori di manipolazione dei bit

Gli operatori di manipolazione dei bit, noti anche come operatori bit per bit, consentono di commutare da 0 a 1 e da 1 a 0 i singoli bit di un intero. Considerate per esempio le seguenti espressioni:

```
$a = 31;
$b = 12;
$c = $a & $b;
```

Con queste espressioni il valore `12` verrà assegnato alla variabile `$c`, in quanto l'operatore `&` effettua un'operazione AND dei bit del primo intero con quelli del secondo. Per comprendere il funzionamento di questo meccanismo, è necessario capire come avviene la memorizzazione degli interi sotto forma di numeri binari in un computer. Considerate la Figura 10.4, che illustra come vengono rappresentati i due numeri 31 e 12 nella notazione binaria a 8 bit.

	128	64	32	16	8	4	2	1
31 =	0	0	0	1	1	1	1	1

	128	64	32	16	8	4	2	1
12 =	0	0	0	0	1	1	0	0

Figura 10.4
Memorizzazione degli interi come numeri binari.

Quando si effettua un'operazione AND su questi due numeri, solo i bit che sono 1 in entrambi i numeri vengono lasciati tali, mentre tutti gli altri vengono impostati a 0. La Figura 10.5 illustra quanto appena affermato.

	128	64	32	16	8	4	2	1
31 =	0	0	0	1	1	1	1	1

	128	64	32	16	8	4	2	1
12 =	0	0	0	0	1	1	0	0

	128	64	32	16	8	4	2	1
31 & 12 =	0	0	0	0	1	1	0	0

Figura 10.5
Operazione AND tra due numeri.

Quando si effettua un'operazione AND con gli interi 31 e 12, si ottiene il valore 12. La Tabella 10.2 elenca gli operatori di manipolazione dei bit disponibili in PHP.

Tabella 10.2 Operatori di manipolazione dei bit

Operatore	Esempio	Descrizione
&	`$a & $b;`	Imposta i bit a 1 quando i bit in entrambi gli operandi sono 1.
¦	`$a ¦ $b;`	Imposta i bit a 1 quando uno dei bit in entrambi gli operandi è 1.
/\	`$a /\ $b;`	Imposta i bit a 1 quando uno dei bit negli operandi è 1, ma non entrambi.
<<	`$a << $b;`	Sposta a sinistra i bit.
>>	`$a >> $b;`	Sposta a destra i bit.
AND	`$a AND $b;`	Imposta i bit a 1 quando i bit in entrambi gli operandi sono 1.
OR	`$a OR $b;`	Imposta i bit a 1 quando uno dei bit in entrambi gli operandi è 1.
XOR	`$a XOR $b;`	Imposta i bit a 1 quando uno dei bit negli operandi è 1, ma non entrambi.

L'operatore di shift a sinistra "<<" sposta a sinistra i bit nell'operando e imposta a zero tutti i bit vacanti. Ogni spostamento a sinistra ha lo stesso effetto di una moltiplicazione per due dell'operando. Lo shift a destra ">>" sposta a destra i bit nell'operando e imposta a zero tutti i bit vacanti. Ogni spostamento a destra equivale a una divisione per due. Lo script che segue dimostra l'utilizzo degli operatori di manipolazione dei bit:

```
<?php
// Espressioni - Esempio 10-2
//-------------------------

echo "(7 & 2) = " . (7 & 2) . '<br>';
echo "(7 ¦ 2) = " . (7 ¦ 2) . '<br>';
echo "(7 ^ 2) = " . (7 ^ 2) . '<br>';
echo "(2 << 2) = " . (2 << 2) . '<br>';
echo "2 >> 1) = " . (2 >> 1);
?>
```

La Figura 10.6 illustra l'output prodotto dallo script precedente.

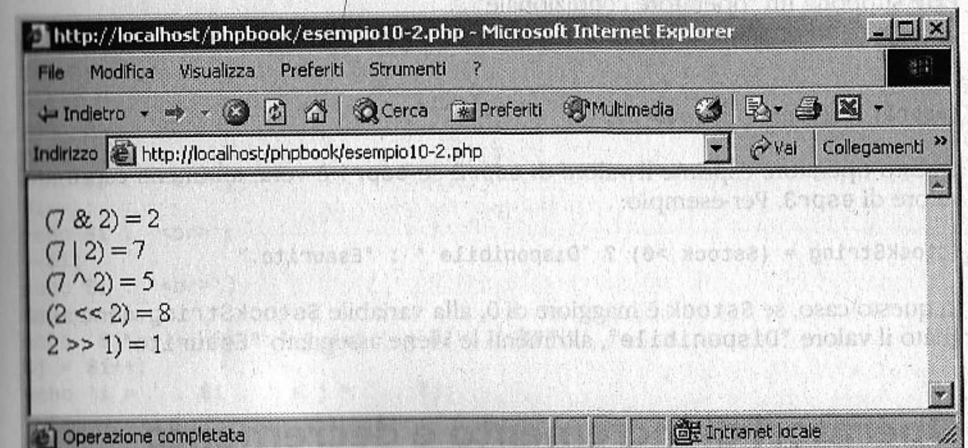


Figura 10.6
Uso degli operatori di manipolazione dei bit.

Operatori di confronto

Gli operatori di confronto vengono utilizzati per confrontare espressioni dal punto di vista logico e aritmetico.
Nella maggior parte dei casi sono impiegati nel flusso delle istruzioni di controllo, che verranno trattate nel Capitolo 8. La Tabella 10.3 illustra gli operatori di confronto supportati da PHP.

Tabella 10.3 Operatori di confronto

Operatore	Esempio	Descrizione
`==`	`$a == $b`	Uguale.
`===`	`$a === $b`	Identico.
`!=`	`$a != $b`	Non uguale.
`!==`	`$a !== $b`	Non identico.
`<`	`$a < $b`	Minore di.
`>`	`$a > $b`	Maggiore di.
`>=`	`$a >= $b`	Maggiore o uguale a.
`<=`	`$a <= $b`	Minore o uguale a.

La differenza tra gli operatori uguale e identico è che gli operatori di uguale e non uguale controllano se i valori dei due operandi sono gli stessi; con gli operatori di identico, invece, viene verificato anche il tipo degli operandi (oltre al valore) per vedere se sono o meno identici.

Operatore condizionale

PHP supporta un "operatore condizionale".
Si tratta di un operatore ternario, in quanto richiede tre operandi, e ha la sintassi seguente:

```
(espr1) ? (espr2) : (espr3);
```

Questo operatore esprime il valore di `espr2` se `espr1` è vera, altrimenti esprime il valore di `espr3`. Per esempio:

```
$stockString = ($stock >0) ? "Disponibile " : "Esaurito."
```

In questo caso, se `$stock` è maggiore di 0, alla variabile `$stockString` viene assegnato il valore `"Disponibile"`, altrimenti le viene assegnato `"Esaurito"`.

Operatori di incremento e decremento

PHP supporta le forme prefissa e suffissa degli operatori di incremento e decremento, elencate nella Tabella 10.4.

Tabella 10.4 Operatori di incremento e decremento

Nome	Esempio	Descrizione
Incremento prefisso	`++$i`	Incrementa `$i` di 1, quindi restituisce `$i`.
Incremento suffisso	`$i++`	Restituisce `$i` e quindi incrementa `$i` di 1.
Decremento prefisso	`--$i`	Decrementa `$i` di 1, quindi restituisce `$i`.
Decremento suffisso	`$i--`	Restituisce `$i` e quindi decrementa `$i` di 1.

Il frammento di script seguente illustra l'utilizzo di questi operatori:

```
$i = 3; // assegna 3 a i
$i++; // i viene incrementata a 4
++$i; // i viene incrementata a 5
$i--; // i viene decrementata a 4
--$i; // i viene decrementata a 3
```

Nel frammento di script precedente non si nota alcuna differenza tra le forme prefissa e suffissa degli operatori. La differenza appare più chiara quando l'operatore di incremento o decremento viene utilizzato in un'espressione di assegnamento. Considerate il frammento seguente:

```
$a = 5; // ad a viene assegnato il valore 5
$b = ++$a; // a viene incrementata e poi assegnata a b
$b = $a++; // a viene assegnata a b e poi incrementata
```

Il risultato di questi assegnamenti è che `b = 6` e `a = 7`. Lo script che segue illustra l'utilizzo di questi operatori:

```
<?php
// Espressioni - Esempio 10-3
//-----------------------

$i =3;
$j =0;
$i++;
echo $i . '<br>';
++ $i;
echo $i . '<br>';
$i-- ;
echo $i . '<br>';
--$i;
echo $i . '<br>';
$j = ++$i;
echo '$i = ' . $i . ' e j = ' . $j . '<br>';
$j = $i++;
echo 'i = ' . $i . ' e j = ' . $j;

?>
```

La Figura 10.7 illustra l'output prodotto dallo script precedente. Osservate che i valori di `$i` e `$j` sono diversi quando vengono utilizzate la forma prefissa e suffissa degli operatori.

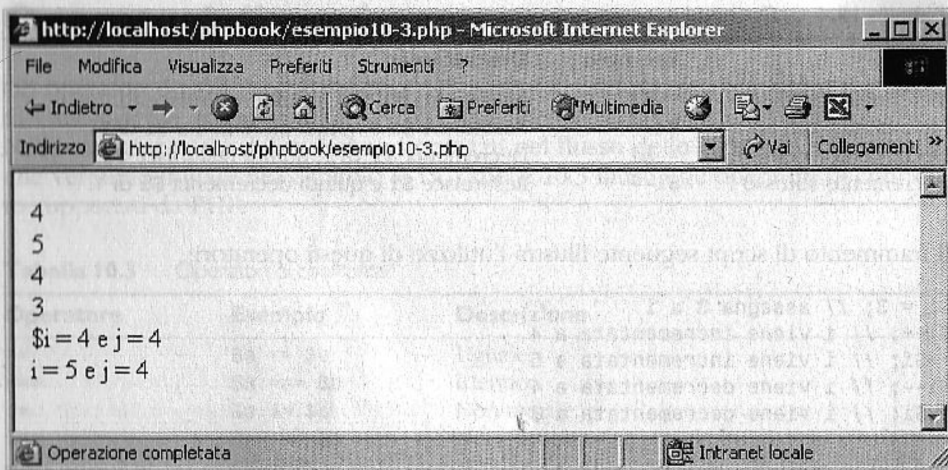


Figura 10.7
Utilizzo degli operatori di incremento e decremento.

Operatori logici

PHP supporta diversi operatori logici, elencati nella Tabella 10.5. Prestate attenzione a non confonderli con gli operatori bit per bit che portano lo stesso nome ma con cui non intrattengono alcuna relazione.

Tabella 10.5 Operatori logici

Operatore	Esempio	Descrizione
And	`$a and $b`	Vero se `$a` e `$b` sono entrambe vere.
Or	`$a or $b`	Vero se sono vere `$a` o `$b`.
Xor	`$a xor $b`	Vero se `$a` o `$b` sono vere, ma falso se sono vere entrambe.
Not	`!$a`	Vero se `$a` non è vera.
And	`$a&&$b`	Vero se `$a` e `$b` sono entrambe vere.
Or	`$a¦¦ $b`	Vero se sono vere `$a` o `$b`.

Gli operatori logici sono utilizzati per combinare i risultati degli operatori di confronto secondo le regole dell'algebra booleana. In PHP un valore 0 viene utilizzato per rappresentare una condizione logica di falsità, mentre il valore 1 indica una condizione logica di verità.
La presenza di due operatori And e Or (AND, &&, OR, ¦¦) nella tabella è dovuta alla priorità degli operatori, di cui ci occuperemo tra breve. Lo script seguente illustra l'utilizzo di questi operatori logici:

```
<?php
// Espressioni - Esempio 10-4
//-----------------------

$i = 3;
$j = 4;

echo (($i < $j) ¦¦ ($j > 5)) . '<br>';
echo (($i < $j) && ($j > 5)) . '<br>';
echo ($i < $j) xor ($j > 5) ;

?>
```

Priorità degli operatori

Quando un'espressione contiene più di un operatore e l'ordine di elaborazione non è stato chiarito con il ricorso alle parentesi, il parser decide come verrà valutata l'espressione in base alla tabella delle priorità degli operatori, illustrata nella Tabella 10.6, che elenca tutti gli operatori in ordine di prorità dal maggiore (ossia gli operatori elaborati per primi) al minore. Alcuni operatori hanno lo stesso ordine di priorità e appaiono sulla stessa riga della tabella. In questi casi è indicata l'associatività di questi operatori, da destra a sinistra o da sinistra a destra, per capire se vengono elaborati a partire dalla parte sinistra o da quella destra dell'espressione.

Tabella 10.6 Priorità degli operatori

Associatività	Operatore
Da destra a sinistra	`[`
Da destra a sinistra	`! ~ ++ --`
Da sinistra a destra	`* / %`
Da sinistra a destra	`+ - .`
Non associativi	`<<= >>=`
Non associativi	`== != === !==`
Da sinistra a destra	`&`
Da sinistra a destra	`^`
Da sinistra a destra	`¦`
Da sinistra a destra	`&&`
Da sinistra a destra	`¦¦`
Da sinistra a destra	`?:`
Da sinistra a destra	`= += -= *= /= .= &= ¦= ^= ~=`
Da destra a sinistra	`print`
Da sinistra a destra	`and`
Da sinistra a destra	`xor`
Da sinistra a destra	`or`
Da sinistra a destra	`,`

Considerate un'espressione come la seguente:

```
$a * $b + $c
```

Essa verrebbe valutata come segue:

```
($a * $b) + $c
```

in quanto l'operatore "*" è più in alto nella tabella delle precedenze, mentre invece:

```
$a/$b * $c
```

benché gli operatori abbiano il medesimo ordine di priorità, verrebbe valutata nel modo seguente:

```
($a/$b) * $c
```

Ciò avviene perché l'associatività degli operatori è da sinistra a destra.

Riepilogo

In questo capitolo sono stati analizzati in modo approfondito espressioni, operandi e operatori. Nel prossimo capitolo verrà introdotta l'istruzione `if` e il suo ruolo di controllo del flusso dello script, che consente di stabilire quali istruzioni devono essere eseguite.

Capitolo 11

Costrutti if e switch

Introduzione

In questo capitolo verrà introdotto il costrutto `if`. Tutti i linguaggi di programmazione utilizzano una forma dell'istruzione `if` per creare condizioni logiche. PHP supporta diverse forme dell'istruzione `if`, delle quali ci occuperemo in queste pagine. Verrà inoltre introdotta l'istruzione `switch`, che equivale a una struttura `if` complessa.

Istruzione if di base

Un'istruzione `if` di base ha una struttura molto semplice, riportata di seguito:

```
if (condizione)
{
        istruzioni da eseguire se la condizione è vera
}
```

La parola "if" è seguita da una coppia di parentesi che contengono l'espressione condizionale che verrà valutata dopo aver raggiunto il costrutto `if`. Se la condizione è vera, le istruzioni comprese tra le parentesi graffe verranno eseguite, altrimenti il programma passerà oltre ignorandole.
Vediamo un esempio di costrutto `if` reale. Supponete di avere due colori e di chiedere a un utente di indicare quello che pensa essere il vostro preferito. Se la risposta corretta è il rosso, una semplice struttura di istruzione `if` per questo esempio apparirà come segue:

```
<?php
// Istruzioni if - Esempio 11-1
//-------------------------

$colour = "rosso";

if ($colour == "rosso")
   {
   echo "Il colore è il rosso";
```

```
}

?>
```

Osservate che, per far funzionare l'esempio precedente, la variabile `$colour = "rosso"` è stata impostata in modo che la condizione `if` sia sempre vera. Se avete una sola istruzione che desiderate eseguire quando l'istruzione `if` è vera, potete omettere le parentesi graffe. Per esempio:

```
<?php
// Istruzioni if - Esempio 11-2
//-------------------------

$colour = "rosso";

if ($colour == "rosso")
      echo "Il colore è il rosso";
?>
```

Possiamo anche modificare nuovamente lo script in modo che visualizzi il valore di `$colour`. A tal fine è sufficiente cambiare l'istruzione `echo` in modo che visualizzi il valore di `$colour`. Possiamo inoltre modificare l'espressione `if` nel modo seguente:

```
if ($colour)
```

Questa istruzione ora significa "se `$colour` contiene un valore diverso da zero". Nel nostro caso il valore "rosso" significherà che la condizione verrà valutata vera, in quanto una stringa non nulla viene valutata 1. Ecco lo script completo:

```
<?php
// Istruzioni if - Esempio 11-3
//-------------------------

$colour = "rosso";

if ($colour)
      echo "Il colore è il $colour";

?>
```

Operatori

I vari operatori di confronto e logici sono stati presentati nel Capitolo 10. Con l'introduzione dell'istruzione `if` disponete, ora, dei mezzi per utilizzarli in modo significativo.
Il frammento di script che segue illustra un'istruzione `if` con l'operatore logico di uguale (==):

```
if  ($a == 1)
{
        echo "VERO";
}
```

Il frammento precedente valuta se la variabile `$a` è impostata a 1, e in tal caso visualizza la parola `"VERO"`. Se la variabile `$a` è impostata a qualunque altro valore, non verrà visualizzato alcunché. Il frammento di script seguente illustra l'utilizzo dell'operatore di non uguale (`!=`):

```
if  ($a != 1)
{
        echo "FALSO";
}
```

In questo esempio, se la variabile `$a` è impostata a 1, non verrà visualizzato alcunché, mentre se è impostata su un qualunque valore diverso da 1, verrà visualizzato il testo `"FALSO"`. È possibile utilizzare una combinazione di operatori di confronto e logici per creare istruzioni complesse. Alcuni esempi sono elencati di seguito.

```
if ($a == 1 or $a == 2){ istruzione }
if ($a == $b and $c == d){ istruzione }
if($a == $b and $c == d){istruzione }
if($a != $b and $c < 5){ istruzione }
if(($a == $b and $c == $d) or ($e < $f)){ istruzione }
if(($a == $b and $c == $d) or ($e < $f or $e === $g)){ istruzione }
if (funzione()){ istruzione }
```

Istruzione if ... else

A volte potreste desiderare che vengano eseguite una o più istruzioni se un'espressione è vera e un altro gruppo di istruzioni se l'espressione è falsa. A tal fine viene utilizzata l'istruzione `else` in congiunzione con l'istruzione `if`. Ecco la sintassi per l'utilizzo dell'istruzione `else`:

```
if ( condizione ) { istruzione } else { istruzione }
```

Considerate lo script seguente:

```
<?php
// Istruzioni if - Esempio 11-4
//-------------------------

$number = 2;
if ($number%2) {
      echo "Il numero $number è dispari";
}
else {
      echo "Il numero $number è pari";
}

?>
```

L'esempio precedente visualizzerà `"Il numero 2 è pari"` poiché la variabile `$number` è un numero pari. Se il numero venisse modificato in 1, verrebbe visualizzato `"Il numero 1 è dispari"`. La condizione `if` funziona utilizzando l'operatore modulo, che in questo caso restituisce il resto della divisione per 2 di `$number`.

Istruzione elseif

L'istruzione `elseif` è una combinazione di `if` ed `else` che opera in modo analogo all'istruzione `else`, in quanto consente l'esecuzione di un'istruzione quando l'espressione `if` è falsa. Tuttavia, a differenza di `else`, `elseif` eseguirà l'istruzione solo se l'espressione `elseif` è vera. Considerate lo script di esempio seguente:

```
<?php
// Istruzioni if - Esempio 11-5
//---------------------------

$number1 = 100;
$number2 = 80;

if ($number1 > $number2) {
        echo "$number1 è maggiore di $number2";
}

elseif ($number1 == $number2) {
        echo "$number1 è uguale a $number2";
}

else {
        echo "$number1 è minore di $number2";
}
?>
```

Questo esempio verifica se il valore di `$number1` è maggiore di `$number2` e, se è vero, visualizza un messaggio per segnalarlo. Se non è vero, i valori vengono confrontati per vedere se sono uguali; se è vero, invece, viene visualizzato il messaggio che segnala questa situazione. Infine, se le due condizioni precedenti sono false, viene visualizzato un messaggio che indica che `$number1` è minore di `$number2`.

Istruzioni complesse

Le istruzioni `if` possono essere combinate per dar vita ad applicazioni molto complesse. È possibile utilizzare una qualunque combinazione di operatori logici o di confronto per controllare l'istruzione `if`. Potete utilizzare `elseif(){..}` o `else{..}` per produrre istruzioni `if` più complesse, oppure ricorrere a istruzioni `if` annidate. L'esempio che segue illustra come verificare un nome utente e una password tramite istruzioni `if`:

```
<?php
//Istruzioni if - Esempio 11-6
//---------------------------

$user = "admin";
$pass = "12345";

$username = "admin";
$password = "123456";

//Verifica del nome utente
if($user == "") {
        echo "Nome utente non specificato!";
}
elseif($user == "$username") {
        echo "Nome utente accettato";
}
else {
        echo "Nome utente non valido";
}

echo "<br />";

//Verifica della password

if($pass == "") {
        echo "Password non specificata!";
}
elseif($pass == "$password") {
        echo "Password accettata";
}
else {
        echo "Password non valida";
}
?>
```

L'esempio precedente utilizza due istruzioni `if`: una per verificare l'esattezza di un nome utente e l'altra per controllare che la password sia corretta, supponendo che il nome utente memorizzato sia "admin" e la sua password sia "123456".
Lo script inizia dichiarando le variabili `$user` e `$pass`, che rappresentano il nome utente e la password forniti dall'utente. Vengono inoltre dichiarate le variabili `$username` e `$password`, che contengono i valori con cui saranno confrontate `$user` e `$pass`:

```
$user = "admin";
$pass = "12345";

$username = "admin";
$password = "123456";
```

La parte successiva dello script controlla se è stato fornito un nome utente e, in tal caso, verifica che il valore immesso sia uguale a quello memorizzato in `$username`:

```
//Verifica del nome utente
if ($user = "") {
        echo "Nome utente non specificato!";
}
elseif ($user == "$username") {
        echo "Nome utente accettato";
}
else {
        echo "Nome utente non valido";
}

echo " <br />";
```

L'ultima parte dello script esegue le stesse operazioni di prima sulla password:

```
// Verifica della password
if ($pass == "") {
        echo "Password non specificata!";
}
elseif ($pass == "$password") {
        echo "Password accettata";
}
else {
        echo "Password non valida";
}
```

Un esempio di output prodotto dallo script precedente è illustrato nella Figura 11.1.

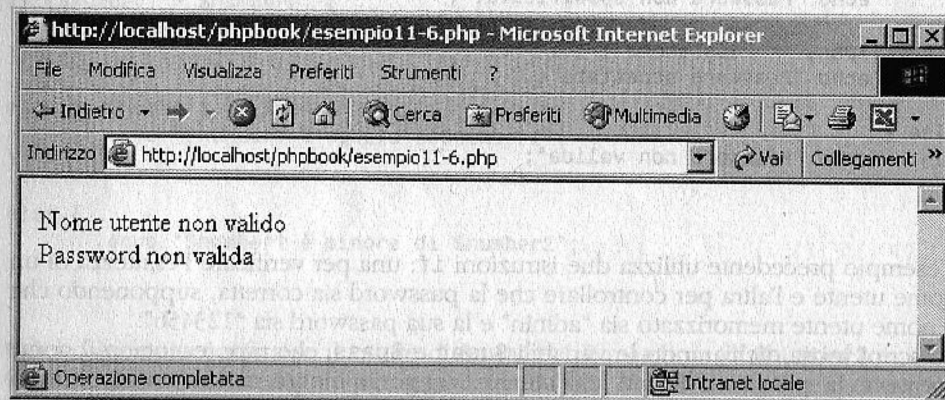


Figura 11.1
Un semplice script di login realizzato con istruzioni if.

Istruzione switch

L'istruzione `switch` è simile a una collezione di istruzioni `if` e viene utilizzata quando desiderate confrontare una variabile con diversi valori ed eseguire diverse istruzioni in relazione al suo valore. Potreste ottenere lo stesso risultato ricorrendo a molte istruzioni `if`, ma c'è il rischio che il codice diventi complesso e difficile da comprendere. La sintassi dell'istruzione `switch` è la seguente:

```
switch (espressione)
{
        case costante_espressione : istruzione
        case costante_espressione : istruzione
        ...
        default : istruzione
}
```

Lo script che segue utilizza un'istruzione `switch` per controllare l'ora locale e mostrare un saluto.

L'ora corrente è memorizzata nella variabile `$time`. Se è 8 viene visualizzato il messaggio `"È ora di alzarsi"`; se è 16, viene visualizzato il messaggio `"È ora di fare merenda"`; se invece è 23, viene visualizzato il messaggio `"È ora di andare a letto"`. Qualunque altra ora produrrà la visualizzazione del testo `"Non è ora di fare nulla"`.

```
<?php
// Istruzioni if - Esempio 11-7
//-------------------------

$time = date ("G");
echo "Sono le ore $time<br />";
switch ($time) {

        case 8 : echo "È ora di alzarsi<br />";
              break;
        case 16 : echo "È ora di fare merenda<br />";
              break;
        case 23 : echo "È ora di andare a letto<br />";
              break;

              default: echo "Non è ora di fare nulla<br />";
}

?>
```

Le istruzioni `break` di questo esempio sono utilizzate per uscire dall'istruzione `switch`. Nel caso in cui fossero omesse, vengono eseguite tutte le istruzioni in tutti i `case` dopo il primo considerato vero, come illustrato di seguito.

```
<?php
// Istruzioni if - Esempio 11-8
//-------------------------

$time = date ("G") ;
echo "Sono le ore $time<br />";
switch ($time) {

        case 11 : echo "Buongiorno!<br />";

        case 16 : echo "Buon pomeriggio!<br />";

        case 23 : echo "Buonanotte!<br />";

}

?>
```

In questo esempio, nel caso in cui siano le ore 11, l'output prodotto è visibile nella Figura 11.2.

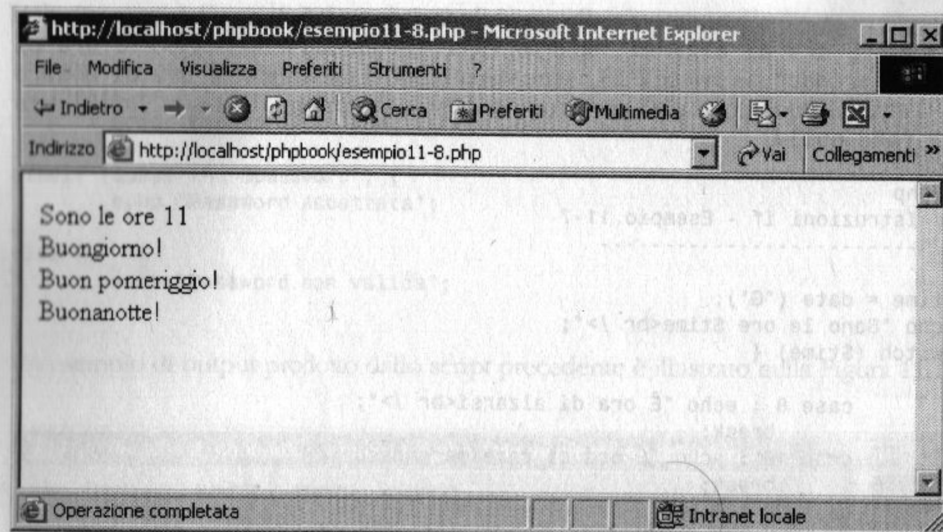


Figura 11.2
Istruzioni switch senza parte break.

Riepilogo

Questo capitolo ha introdotto le istruzioni `if` e `switch` illustrando come possano essere impiegate per controllare quali istruzioni vengono eseguite. Si è visto inoltre come gli operatori logici e relazionali introdotti nel capitolo precedente possano essere utilizzati con queste istruzioni. Nel prossimo capitolo verranno presentate altre strutture di controllo, i cicli.

Capitolo 12

Cicli

Introduzione

Come in qualunque linguaggio di programmazione, i cicli costituiscono una parte essenziale e vengono utilizzati per consentire l'esecuzione ripetuta di un blocco di codice. Esistono quattro tipi di cicli: `for`, `while`, `do while` e `for each`, ciascuno dei quali prevede impieghi specifici. Tutti i tipi di ciclo sono controllati da espressioni e condizioni. In questo capitolo verranno analizzati i primi tre tipi di ciclo, mentre il costrutto `for each` verrà illustrato nel Capitolo 14, in quanto si tratta di un tipo di ciclo utilizzato soltanto per accedere al contenuto di un array. Per cominciare sarà preso in esame il ciclo `while`.

Ciclo while

I cicli `while` presentano una struttura poco complessa e sono molto comuni in PHP. Il significato di un'istruzione `while` è semplice: essa indica all'interprete PHP di eseguire ripetutamente le istruzioni annidate nel ciclo fino a quando l'espressione `while` viene valutata `TRUE`. Il formato del ciclo `while` è il seguente:

```
while (espressione) {istruzione}
```

A ogni iterazione il valore dell'espressione viene verificato all'inizio del ciclo; se l'espressione `while` viene valutata `FALSE` sin dall'inizio, le istruzioni annidate non saranno eseguite nemmeno una volta. È possibile utilizzare le parentesi graffe per racchiudere diverse istruzioni che devono essere eseguite nel ciclo. Lo script seguente illustra un semplice ciclo `while`:

```
<?php
// Cicli - Esempio 12-1
//-----------------

$a = 1;
while ($a<=10)

{
    echo "Questo è il numero $a<br>";
```

```
    $a++;
}
?>
```

In questo esempio la variabile `$a` è impostata a 1. L'espressione del ciclo verifica se il valore di `$a` è minore o uguale a 10 e in tal caso il contenuto del ciclo viene elaborato. In questo esempio viene utilizzata un'istruzione `echo` per visualizzare il valore della variabile `$a`. Inoltre il valore della variabile `$a` viene incrementato di 1 a ogni esecuzione del contenuto del ciclo. Alla fine il valore di `$a` sarà pari a 11, quindi lo script terminerà. L'output di questo script è illustrato nella Figura 12.1.

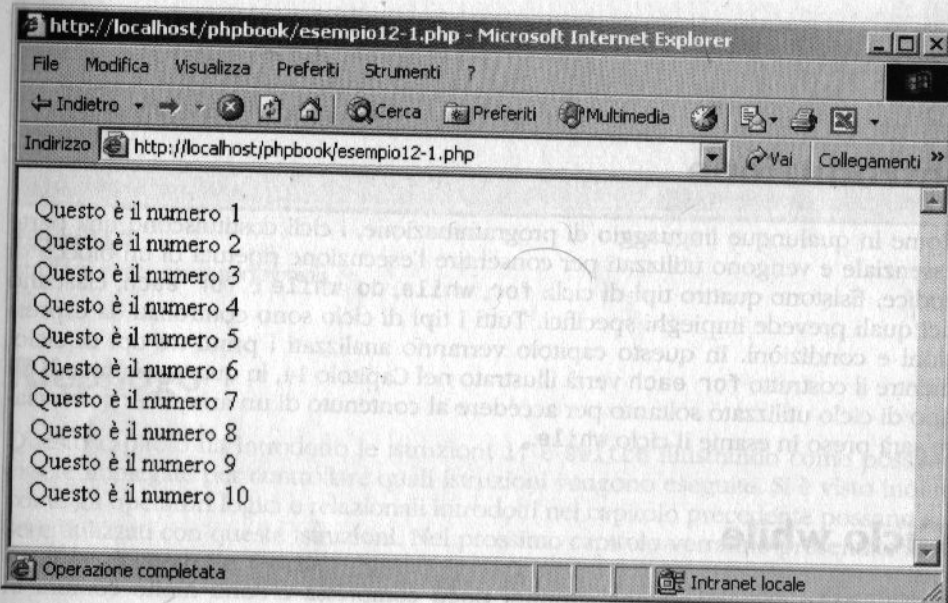


Figura 12.1
Un semplice ciclo while.

I cicli `while`, nella maggior parte dei casi, vengono utilizzati con una variabile intera che viene incrementata o decrementata a ogni iterazione del ciclo. L'espressione del ciclo viene verificata ogni volta per determinare se il ciclo dev'essere eseguito ancora una volta oppure se il controllo deve passare all'istruzione che segue il ciclo (in questo caso la fine dello script). Se una variabile non viene né incrementata né decrementata nel ciclo, quest'ultimo non avrà termine, e lo stesso accadrà allo script PHP. Per esempio:

```
<?php
// Cicli - Esempio 12-2
//-----------------

$a = 1;
while ($a<=10)

    echo "Questo è il numero $a<br>";
?>
```


Consigliamo di non provare a eseguire questo script nel browser, poiché il server Web potrebbe non rispondere più. Questo script, tuttavia, mostra un'utile caratteristica dei cicli che racchiudono al loro interno un'unica istruzione, ossia la possibilità di omettere la coppia di parentesi graffe di apertura e chiusura.

Ciclo do while

Il ciclo `do while` è simile al ciclo `while`; la differenza si nota nel momento in cui viene verificata la veridicità dell'espressione del ciclo. In un ciclo `while` l'espressione del ciclo viene verificata all'inizio; pertanto, se l'espressione è falsa, nessuna delle istruzioni del ciclo verrà eseguita. Nel caso di un ciclo `do while`, invece, l'espressione viene valutata alla fine del ciclo, dopo che le istruzioni sono state eseguite almeno una volta. Questo è il motivo per cui un ciclo `do while` è noto come ciclo iterativo "uno o molti", in quanto le istruzioni al suo interno vengono eseguite almeno una volta, e forse molte volte. Un ciclo `while` è noto, invece, come ciclo "zero, uno o molti" poiché le istruzioni potrebbero non essere eseguite. Il formato del ciclo `do while` è il seguente:

```
do {istruzione} while (espressione);
```

Lo script seguente illustra un semplice ciclo `do while`:

```
<?php
// Cicli - Esempio 12-3
//-----------------

$num = 0;
do {
echo "Questo è un ciclo do-while";
} while ($num > 0);

?>
```

L'esempio precedente illustra un ciclo che non visualizzerebbe alcunché se venisse implementato con il costrutto `while`, in quanto l'espressione viene valutata subito falsa. Dato invece che un ciclo `do while` elabora sempre il contenuto del ciclo almeno una volta (poiché l'espressione viene verificata alla fine e non all'inizio del ciclo), il messaggio `"Questo è un ciclo do-while"` viene visualizzato.

Ciclo for

Un altro tipo di ciclo molto diffuso è `for`, anche se a prima vista appare molto più complesso dei cicli `while` e `do while`, in quanto il costrutto incorpora tre espressioni. La prima espressione viene utilizzata per impostare un punto di inizio del ciclo, la seconda è l'espressione che controlla quante volte verrà ripetuto il ciclo, mentre la terza è quella utilizzata normalmente per incrementare la variabile utilizzata per controllare il numero di iterazioni. Il formato del ciclo `for` è il seguente:

```
for (espressione1; espressione2; espressione3) {istruzione}
```

È possibile utilizzare le parentesi graffe per racchiudere diverse istruzioni da esegui-

re nel ciclo, proprio come avviene nei cicli `while` e `do while`. L'esempio che segue illustra un breve ciclo `for` utilizzato per generare dieci numeri:

```
<?php
// Cicli - Esempio 12-4
//-----------------

for ($count=0; $count<=10; $count++)
{
        echo "$count ";
}
?>
```

In questo esempio il valore iniziale di `$count` viene definito come 0 dalla prima espressione, che indica al ciclo di partire da 0. La seconda espressione è la condizione di terminazione del ciclo e verifica se `$count` è minore o uguale a 10. Se è così, il ciclo continua. L'ultima espressione incrementa la variabile `$count` di 1 a ogni iterazione del ciclo. L'output di questo script è illustrato nella Figura 12.2.

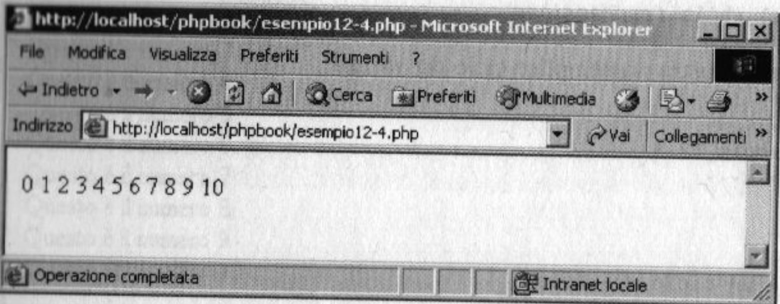


Figura 12.2
Utilizzo di un ciclo for per generare dieci numeri.

Spesso i cicli possono essere utilizzati per generare molto codice HTML, in particolare tabelle. Nell'esempio che segue vedremo come un ciclo `for` possa dar vita a una tabella con sei righe e due colonne:

```
<?php
// Cicli - Esempio 12-5
//-----------------

?>

<html>
<body>
<table width "40%" border="1">

<?
for ($c=0;$c<6;$c++)
{
    echo "<tr>";
    echo "<td>Colonna 1 Riga $c</td>";
    echo "<td>Colonna 2 Riga $c</td>";
    echo "</tr>";
}
?>
</table>
</body>
</html>
```

L'output di questo script è illustrato nella Figura 12.3.

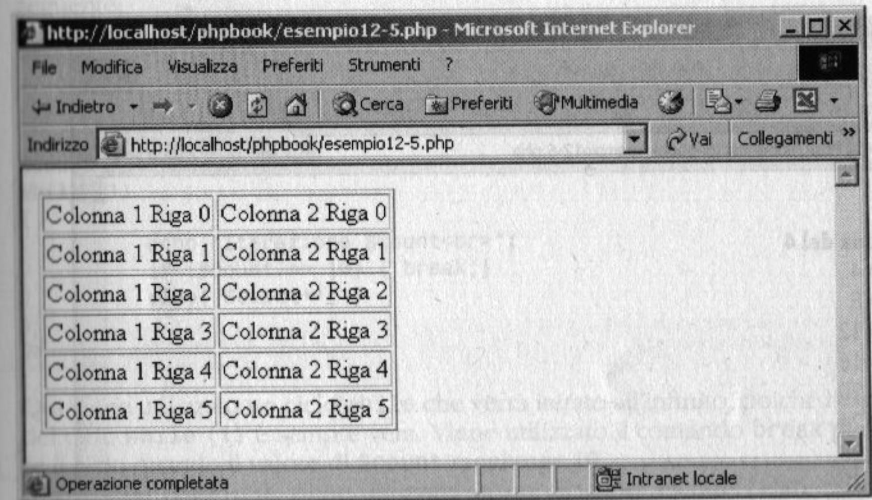


Figura 12.3
Utilizzo di un ciclo for per creare una tabella con numeri di colonna e di riga.

Cicli annidati

Un ciclo annidato è un ciclo interno ad un altro. La prima iterazione del ciclo più esterno comporta l'elaborazione per intero del ciclo interno; poi ha inizio la seconda iterazione del ciclo esterno, che elabora nuovamente il ciclo interno, e così via. I cicli annidati costituiscono uno strumento di programmazione molto potente, in grado di produrre output complessi con un numero ridotto di istruzioni.
È possibile annidare una qualunque combinazione di cicli di diverso tipo. Il codice che segue illustra un ciclo `while` annidato in un ciclo `for`.

```
<?php
// Cicli - Esempio 12-6
//-----------------

for ($y=4; $y<6; $y++)
{
        echo "<br><b> Tabellina del $y </b><br>";

        $x=1;
        while ($x<13)
{
                $result = $x*$y;
                echo "$y x $x = $result <br>";
```

```
    }
}
?>
```

L'esempio crea le tabelline del quattro e del cinque. Il ciclo `for` esterno viene utilizzato per creare ciascuna tabellina, mentre il ciclo `while` interno serve eseguire i calcoli. L'output prodotto è illustrato nella Figura 12.4.

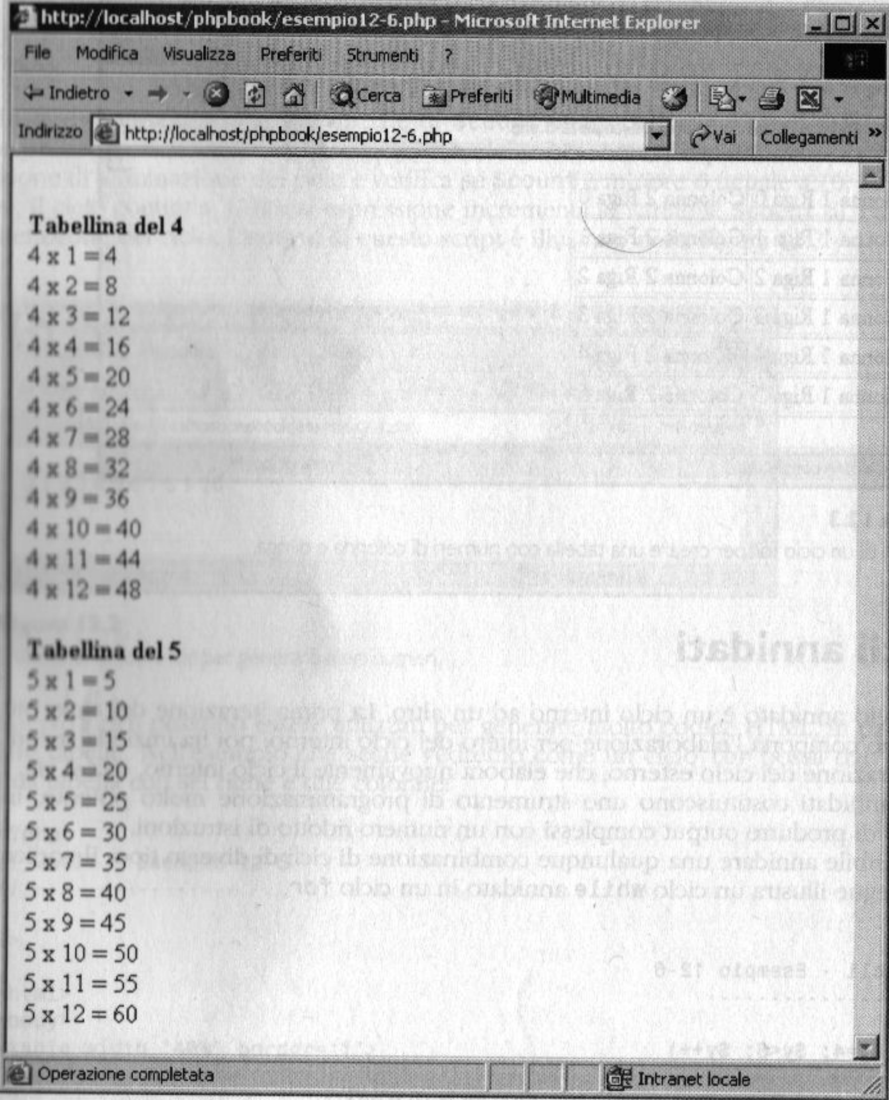


Figura 12.4
Le tabelline create con i cicli annidati.

Interruzione o continuazione forzata di un ciclo

Il comando `break` può anche essere utilizzato per porre termine all'esecuzione di un ciclo `while`, `do while` o `for`; può inoltre accettare un argomento di tipo numerico, che indica da quanti cicli annidati è necessario uscire. Considerate l'esempio seguente:

```
<?php
// Cicli - Esempio 12-7
//-----------------

$count = 0;
while (1)
{
        echo "Iterazione $count<br>";
        if ($count == 10) { break;}
        else{ $count++; }
}
?>
```

Qui viene utilizzato un ciclo `while` che verrà iterato all'infinito, poiché l'espressione del ciclo `while (1)` è sempre vera. Viene utilizzato il comando `break` per terminare il ciclo quando il valore di `$count` raggiunge 10.
L'istruzione `continue` viene utilizzata in un ciclo per evitare l'elaborazione delle istruzioni rimanenti dell'iterazione corrente, passando all'iterazione successiva. Considerate l'esempio seguente:

```
<?php
// Cicli - Esempio 12-8
//-----------------

for ($count=1; $count<20; $count++)
{
        if ($count % 2) { continue; }
        echo "$count è pari <br>";
}
?>
```

Questo script viene iterato da 1 a 19 e visualizza solo i numeri dispari. Infatti, nel caso in cui il valore di `$count` sia pari, viene eseguita l'istruzione `continue`, che non porta all'esecuzione dell'istruzione `echo`. L'output di questo script è illustrato nella Figura 12.5.

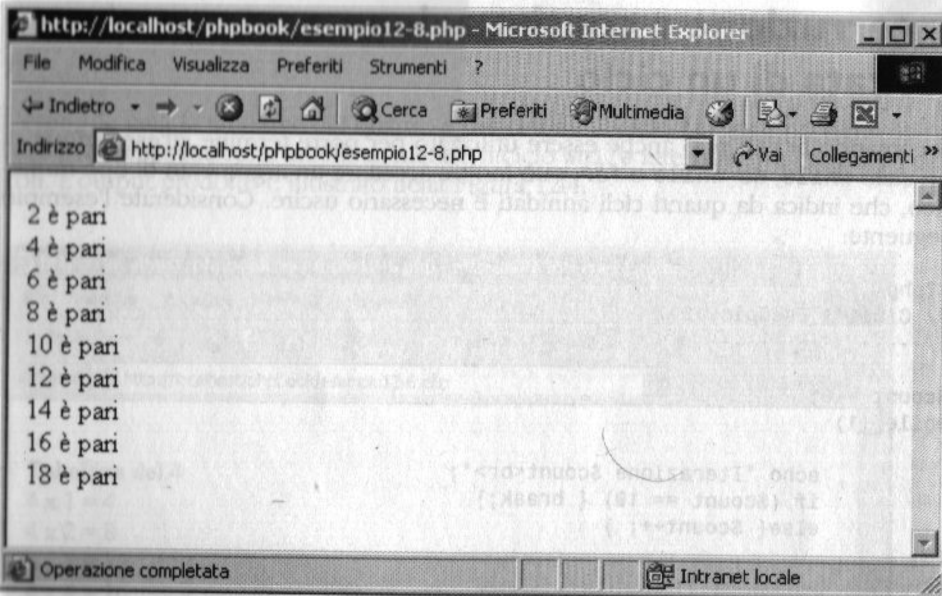


Figura 12.5
Utilizzo dell'istruzione continue.

Riepilogo

In questo capitolo è stato introdotto il concetto di flusso di controllo attraverso i cicli. Avete visto che i cicli possono essere impiegati per ripetere molte volte una sezione del codice e sono quindi utili al programmatore, in quanto riducono il lavoro di codifica necessario. Avete visto come i cicli possano contenere un numero variabile di istruzioni PHP e possano persino includere altri cicli. Nel prossimo capitolo vedrete come potete iniziare a dividere e strutturare il codice attraverso funzioni e file inclusi.

Capitolo 13

Funzioni e file inclusi

Introduzione

Questo capitolo introduce il concetto di funzioni. PHP è dotato di una libreria che contiene centinaia di funzioni già scritte e pronte all'uso. Le funzioni semplificano notevolmente il lavoro dei programmatori espandendo il nucleo fondamentale del linguaggio e mettendo a disposizione funzionalità avanzate. Tuttavia, per quanto possa essere valida una libreria di funzioni, essa non conterrà mai tutte le funzioni che occorrono. In questi casi dovrete crearne di vostre. Potreste anche voler dividere il codice in file PHP separati e includerli insieme per dar vita a un unico grande script PHP. Questo capitolo spiega come fare.

Che cos'è una funzione

Le funzioni sono blocchi di codice che vengono elaborati solo quando la funzione viene chiamata. Se la funzione non viene chiamata, il codice al suo interno non verrà mai eseguito. Una funzione può essere chiamata (a volte si dice anche "invocata") da un punto qualunque dello script. Una volta eseguita la funzione, il flusso di controllo del programma ritorna al punto dello script immediatamente successivo alla chiamata.

Funzioni definite dall'utente

Le funzioni definite dall'utente sono blocchi di script che potreste voler chiamare più volte. Le funzioni eliminano la necessità di duplicare il medesimo blocco di script, in quanto una funzione può essere chiamata dall'interno di un semplice ciclo. Ciò riduce le dimensioni dello script e rende più produttivo il programmatore. Le funzioni conferiscono una maggiore leggibilità agli script e pertanto ne agevolano la manutenzione.
Una funzione può essere definita con la sintassi seguente:

```
funtion nome (arg_1, arg_2, ..., arg_n)
{
```

```
script
.
.
    return valore;
}
```

Le funzioni hanno sempre un nome univoco (in modo da poter identificare quale funzione utilizzare). Presentano anche codice racchiuso tra parentesi graffe che indicano l'inizio e la fine della funzione. Alcune funzioni hanno argomenti, ossia variabili che vengono passate alla funzione e determinano ciò che la funzione farà. Alcune funzioni, inoltre, restituiscono un singolo valore variabile.

Creazione e chiamata di una funzione

L'esempio che segue illustra una semplice funzione che crea una tabella HTML. Non vengono definiti argomenti e non viene restituito alcun valore al termine dell'esecuzione della funzione, ma verrà creata una tabella. Questa è la struttura più semplice di funzione definita dall'utente e viene utilizzata raramente:

```
<?php

//Funzioni e file inclusi - Esempio 13-1
//----------------------------

function create_table()
{
 echo" <table border '1' bgcolor='#DBDBDB' width='394' height='121'> ";
 echo" <tr>";
 echo"   <td>cella 1</td> ";
 echo"   <td>cella 2 </td> ";
 echo" </tr> ";
 echo" </table> ";
 echo"<br />";
}

// Chiama la funzione
create_table();

?>
```

Lo script definisce una funzione denominata `create_table`. Il codice contenuto nella funzione è quello racchiuso tra le parentesi graffe {} ed è costituito da una serie di istruzioni `echo`:

```
function create_table()
{
 echo" <table border '1' bgcolor='#DBDBDB' width='394' height='121'> ";
 echo" <tr>";
 echo"   <td>cella 1</td> ";
 echo"   <td>cella 2 </td> ";
 echo" </tr> ";
 echo" </table> ";
 echo"<br />";
}
```


Per eseguire la funzione è necessario chiamarla, e ciò avviene con la riga seguente:

```
create_table();
```

Questa istruzione chiama la funzione senza passarle argomenti. L'output di questa funzione è illustrato nella Figura 13.1.

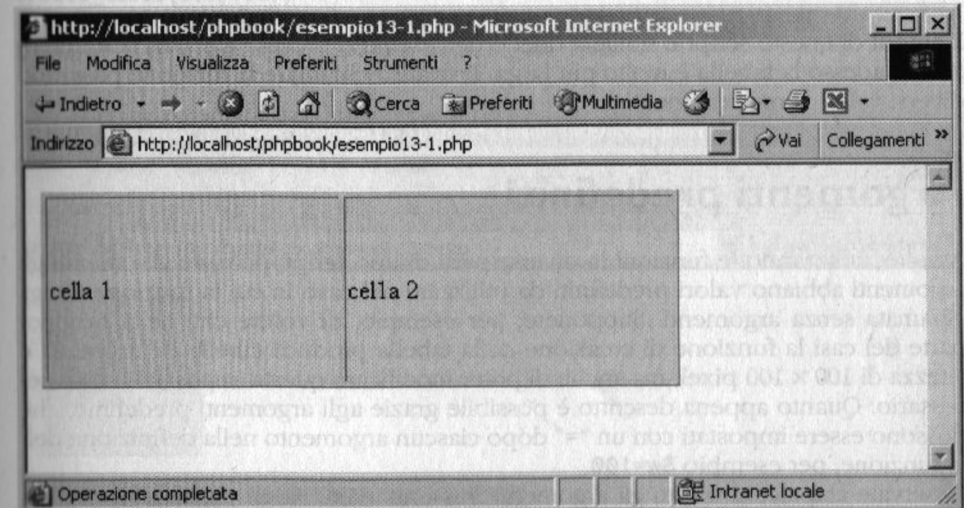


Figura 13.1
L'output di una semplice funzione.

Creazione di una funzione con argomenti

Gli argomenti sono utilizzati per controllare l'output e il comportamento della funzione. Nell'esempio precedente è stata creata una tabella HTML statica e nient'altro. Ma se voleste modificare la dimensione della tabella in base alle vostre necessità? Questo è possibile utilizzando argomenti per controllare la larghezza e l'altezza della tabella. Nell'esempio che segue l'argomento `$w` definirà la larghezza della tabella, mentre `$h` ne determinerà l'altezza. Gli argomenti sono separati da virgole (,) e non ci sono limiti alla quantità utilizzabile in una funzione. Nel caso in cui l'argomento sia uno solo, non sono necessarie virgole. Ecco uno script di esempio che utilizza una funzione con argomenti:

```
<?php

//Funzioni e file inclusi - Esempio 13-2
//----------------------------

function create_table($w, $h)
{
 echo" <table border '1' bgcolor='#DBDBDB' width='$w' height='$h'> ";
 echo" <tr>";
 echo"   <td>cella 1</td> ";
 echo"   <td>cella 2 </td> ";
```

```
echo" </tr> ";
echo" </table> ";
echo"<br />";
}
//Chiama la funzione passando i valori
create_table(500,100);

?>
```

L'output di questo script è simile a quello dell'esempio precedente, con la differenza che adesso la tabella è molto più larga. Provate a chiamare la funzione passando diversi valori come argomenti.

Argomenti predefiniti

A volte, utilizzando le funzioni in diverse parti di uno script, potreste volere che gli argomenti abbiano valori predefiniti da utilizzare nel caso in cui la funzione venga chiamata senza argomenti. Supponete, per esempio, di volere che nella maggior parte dei casi la funzione di creazione della tabella produca tabelle di larghezza e altezza di 100 × 100 pixel, ma anche di poter modificare questa impostazione se necessario. Quanto appena descritto è possibile grazie agli argomenti predefiniti, che possono essere impostati con un "=" dopo ciascun argomento nella definizione della funzione, per esempio `$w=100`.
Osservate che alcuni o tutti gli argomenti possono avere valori predefiniti, che devono essere separati da virgole. Lo script seguente fornisce un esempio:

```
<?php

//Funzioni e file inclusi - Esempio 13-3
//--------------------------

function create_table($w=100, $h=100)
{
 echo" <table border '1' bgcolor='#DBDBDB' width='$w' height='$h'> ";
 echo" <tr>";
 echo"   <td>cella 1</td> ";
 echo"   <td>cella 2 </td> ";
 echo" </tr> ";
 echo" </table> ";
 echo"<br />";
}
//Chiama la funzione senza valori
create_table();

//Chiama la funzione passando i valori
create_table(300,100);

?>
```

In questo esempio viene utilizzata la stessa funzione `create_table()`, ma gli argomenti `$w` e `$h` hanno ciascuno il valore predefinito 100:

```
function create_table($w=100, $h=100)
```

Ora è possibile chiamare la funzione senza argomenti:

```
create_table();
```

Potete però sovrascrivere gli argomenti predefiniti impostando un nuovo valore nella chiamata a funzione:

```
create_table(300,100);
```

L'output di questo script è illustrato nella Figura 13.2.

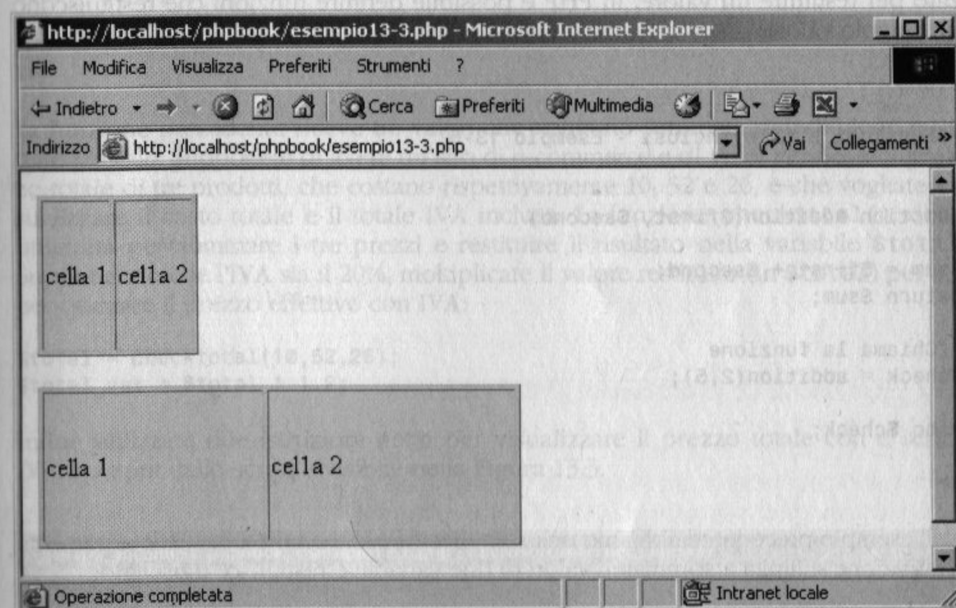


Figura 13.2
Argomenti predefiniti.

Restituzione di valori

L'esecuzione del codice interno alle funzioni si interrompe quando esse incontrano la parola chiave `return`:

```
<?php

//Funzioni e file inclusi - Esempio 13-4
//--------------------------

function test()
{
 echo "Questa è una prova";
 return;
```

```
echo "Questa è un'altra prova";
}

test() ;

?>
```

In questo esempio l'output prodotto è "`Questa è una prova`", poiché l'istruzione `return` viene prima di raggiungere la seconda istruzione `echo`.
Questa parola chiave non sarebbe molto utile se venisse utilizzata solo in questo modo, quindi ecco un altro possibile impiego. Il comando `return` può essere utilizzato per restituire un valore. In PHP è possibile definire funzioni che restituiscono un singolo valore:

```
<?php

//Funzioni e file inclusi - Esempio 13-5
//-----------------------------

function addition($first, $second)
{
$sum = $first + $second;
return $sum;
}
//Chiama la funzione
$check = addition(2,5);

echo $check;

?>
```

Nell'esempio precedente abbiamo una semplice funzione che utilizza due argomenti. La funzione si limita a sommare i valori di due variabili e restituisce il risultato:

```
function addition($first, $second)
{
$sum = $first + $second;
return $sum;
}
```

Dato che questa funzione restituisce un valore, è necessario "catturarlo", altrimenti andrà perduto; a tal fine lo si può memorizzare in una variabile, che è stata chiamata `$check`:

```
$check = addition(2,5);
```

Tutto ciò che la funzione restituisce viene memorizzato nella variabile `$check`. Ricorrendo a un'istruzione `echo` è possibile, inoltre, osservare ciò che è stato restituito dalla funzione, in questo caso il valore 7. Lo script che segue mostra un esempio di valore restituito molto più utile:

```
<?php

//Funzioni e file inclusi - Esempio 13-6
//-----------------------------
```



```
function checktotal($a, $b, $c)
{
 $sum = $a + $b + $c;
 return $sum;
}

//Chiama la funzione
$total = checktotal(10,52,26);
$total_vat = $total * 1.2;

echo "Totale senza IVA: <b>$total</b>";
echo "<br>";
echo "Totale con IVA 20%: <b>$total_vat</b>";

?>
```

La funzione precedente riceve tre valori che vengono sommati, quindi viene restituito il totale. Supponete di avere un sito di e-commerce e di voler verificare il prezzo totale di tre prodotti, che costano rispettivamente 10, 52 e 26, e che vogliate visualizzare il costo totale e il totale IVA inclusa. La funzione `checktotal()` viene utilizzata per sommare i tre prezzi e restituire il risultato nella variabile `$total`. Supponendo che l'IVA sia il 20%, moltiplicate il valore restituito (in `$total`) per 1,2 per ottenere il prezzo effettivo con IVA:

```
$total = checktotal(10,52,26);
$total_vat = $total * 1.2;
```

Infine utilizzate due istruzioni `echo` per visualizzare il prezzo totale con e senza IVA; l'output dello script è visibile nella Figura 13.3.

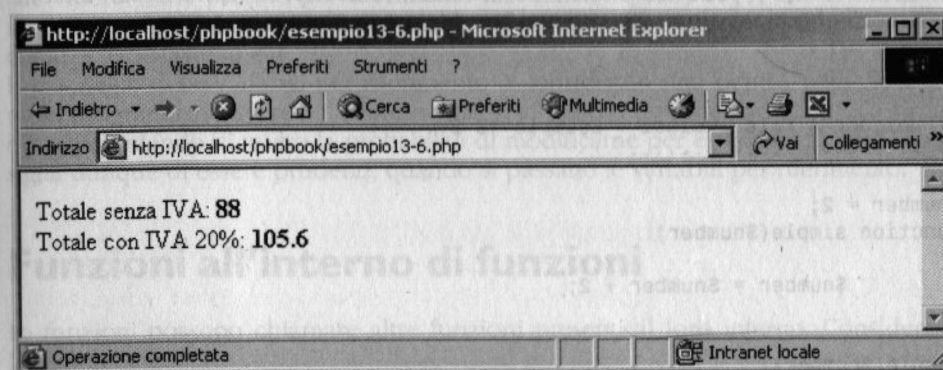


Figura 13.3
Utilizzo dei valori restituiti.

Restituzione di più valori

In precedenza abbiamo accennato al fatto che in PHP le funzioni non possono restituire più di un valore. Per la precisione si dovrebbe dire che possono solo restituire un tipo singolo. Per aggirare il problema della restituzione di un singolo valore

è possibile utilizzare gli array. Nel prossimo capitolo vedrete come definire e utilizzare gli array, ma per ora ecco un semplice esempio di funzione che restituisce un array. Non preoccupatevi se non comprendete la sintassi degli array; verrà spiegata tra breve:

```
<?php

//Funzioni e file inclusi - Esempio 13-7
//---------------------------

function three_names()
{
  $names = array("Simon", "Mike", "Liz");
  return $names;
}
//Chiama la funzione
$myNames = three_names();

echo "Primo nome: " . $myNames[0] ;
echo "<br />Secondo nome: " . $myNames[1];
echo "<br />Terzo nome: " . $myNames [2] ;

?>
```

Argomenti passati per valore

Quando si passa un argomento a una funzione, il suo valore viene automaticamente "passato per valore"; ciò significa che viene passata solo una copia della variabile. Quindi se quel valore viene modificato nella funzione, il valore della variabile esterna alla funzione rimane invariato. Considerate l'esempio seguente:

```
<?php

//Funzioni e file inclusi - Esempio 13-8
//---------------------------

$number = 2;
function simple($number)
{
      $number = $number + 2;
}

simple($number);
echo $number;
?>
```

Questo esempio dichiara una variabile `$number` e le assegna il valore 2. Il valore viene passato alla funzione `simple()` e viene poi cambiato in 4. Dopo la chiamata alla funzione, viene visualizzato il valore di `$number`, che è 2. In un certo senso si potrebbe dire che la funzione non ha avuto alcun effetto sul valore della variabile. Il motivo per cui la variabile sembra tornare al valore originale è che alla funzione è stata passata una copia della variabile, ed è questa copia che viene modificata, non l'originale. Quando la funzione ha completato la propria esecuzione, la variabile originale all'esterno della funzione esiste ancora ed è immutata.

Argomenti passati per riferimento

Potreste voler passare una variabile a una funzione "per riferimento". L'utilizzo delle variabili per riferimento è stato introdotto nel Capitolo 6. Per passare una variabile per riferimento a una funzione è sufficiente anteporre il carattere "&" al nome della variabile. L'utilizzo delle variabili per riferimento consente alla funzione di modificare il valore originale della variabile. Considerate l'esempio seguente:

```
<?php

//Funzioni e file inclusi - Esempio 13-9
//---------------------------

$variable = 2;

function simple(&$variable)
{
        $variable = $variable + 2;
}
simple($variable);
echo $variable;

?>
```

Questo script dichiara una variabile `$number`, alla quiale viene assegnato il valore 2. tale valore viene passato per riferimento alla funzione `simple()`, che lo modifica in 4. Dopo la chiamata alla funzione viene visualizzato il valore di `$number`, che come ci aspettavamo è 4.
Il passaggio delle variabili per riferimento è considerato una pratica pericolosa da molti programmatori (anche professionisti), in quanto estende l'ambito della variabile, aumentando di molto la probabilità di modificarne per errore il valore. Si consiglia dunque di essere prudenti, quando si passano le variabili per riferimento.

Funzioni all'interno di funzioni

Le funzioni possono chiamare altre funzioni presenti al loro interno. Considerate l'esempio seguente:

```
<?php
//Funzioni e file inclusi - Esempio 13-10
//---------------------------

function decimal ($num)
{
        $num = round($num, 2);
        return $num;
}
```

```
function multiply(&$number)
{
        $number = $number * 3.14;
        echo $number . "<br />";
        $number = decimal($number);
        echo = $number;
}

$number = 5.2234;
multiply($number);

?>
```

In questo script viene definito un valore in virgola mobile che viene passato alla funzione `multiply()`:

```
$value = 5.2234;
multiply($number};
```

La funzione moltiplica il numero per 3,14 e quindi visualizza il suo valore corrente:

```
$number = $number * 3.14;
echo $number . "<br />;
```

Poi chiama la funzione `decimal()` passandole il valore di `$number` e supponendo che il valore verrà restituito e infine visualizzato:

```
$number = decimal($number};
echo = $number;
```

La funzione `decimal()` riceve la variabile e con la funzione predefinita `round()`, arrotonda il valore alla seconda cifra decimale e restituisce il risultato:

```
function decimal ($num)
{
        $num = round($num, 2};
        return $num;
}
```

L'output prodotto da questo script è il seguente:

```
16.401476
16.4
```

Funzioni ricorsive

Una funzione ricorsiva è una funzione che chiama se stessa. Si tratta di una situazione complessa, ma utile nella valutazione di alcuni tipi di funzioni matematiche. Potreste trovare costrutti di questo tipo in alcune strutture dinamiche di gestione dei dati, come liste collegate o alberi binari.

Il codice che segue presenta un esempio di utilizzo di una funzione ricorsiva per determinare se un numero è una potenza di 2.

```
<?php
//Funzioni e file inclusi - Esempio 13-11
//-----------------------------

function is_power_of_two($n)
{
   if ($n == 1)
   {
      echo "sì";
   }
   elseif ($n%2 == 1)
   {
      return "no";
   }
   else
   {
      $n /= 2;
      return is_power_of_two($n);
   }
}
echo is_power_of_two(256);
?>
```

In questo esempio il numero 256 viene controllato per vedere se è o meno una potenza di 2 passandolo alla funzione `is_power_of_two()`:

```
echo is_power_of_two(256);
```

Quando la funzione viene chiamata, un'istruzione `if` verifica se il numero è uguale a 1 e, in tal caso, viene visualizzato il valore `"sì"`; in caso contrario viene utilizzata un'altra istruzione `if`, per verificare se la divisione per 2 del valore corrente `$n` è uguale a 1. Se è vero, viene visualizzato il valore `"no"`:

```
if {$n == 1)
   {
      echo "sì";
   }

elseif ($n%2 == 1)
   {
      return "no";
   }
```

Se il numero non soddisfa nessuna di queste due istruzioni viene diviso per 2, e la funzione chiama se stessa con l'istruzione `return` *`funzione`*`()`:

```
else
   {
      $n /= 2;
      return is_power_of_two($n);
   }
```

include, include_once

Finora si è visto come creare funzioni che svolgono operazioni specifiche. Le funzioni consentono di dividere il codice in porzioni gestibili. Un altro modo per suddividere il codice consiste nel separare uno script PHP di grandi dimensioni in diversi script più piccoli. Questa tecnica consente di condividere e riutilizzare in modo più agevole gli script. Considerate lo script seguente:

```
<?php
//Funzioni e file inclusi - Esempio 13-12
//-------------------------

$firstname = "Simon";
$surname = "Stobart";

?>
```

Effettivamente non è molto interessante, poiché si limita a definire due variabili; tuttavia è possibile includere questo file in un altro script tramite l'istruzione `include`:

```
include (nomefile);
```

Per esempio:

```
<?php
//Funzioni e file inclusi - Esempio 13-13
//---------------------------

echo $firstname;
include ("esempio13-12.php");
echo "<br />" . $surname;
?>
```

Lo script precedente utilizza l'istruzione `include` per "copiare" lo script del file esempio 13-12.php nel punto dello script in cui compare l'istruzione `include`. Questo produce la visualizzazione del messaggio seguente:

```
Notice: Undefined variable: firstname in
c:\inetpub\wwwroot\phpbook\esempio13-13.php on line 6

Stobart
```

Questo avviso viene generato poiché il valore della variabile `$firstname` non è noto quando si raggiunge la prima istruzione `echo`. Sarebbe sufficiente spostare l'istruzione `include` prima della prima istruzione `echo` per eliminare il messaggio. Nel caso in cui sia impossibile trovare il file incluso, viene generato un avviso, ma l'elaborazione dello script continua. L'istruzione `include_once()` opera allo stesso modo di `include()`, con l'unica differenza che lo script contenuto in un file viene incluso una sola volta nell'esecuzione dello script. Il formato dell'istruzione è il seguente:

```
Include_once (nomefile);
```


require, require_once

Queste istruzioni funzionano esattamente allo stesso modo di `include` e `include_once`; l'unica differenza è nella modalità di gestione degli errori. Nel caso sia impossibile individuare un file, viene generato un errore irreversibile e l'esecuzione dello script termina.

Riepilogo

Questo capitolo ha introdotto le funzioni, mostrando come vengono create e come sia possibile passare a esse variabili sia per valore sia per riferimento. Si è visto inoltre che è possibile includere script PHP separati in uno script da eseguire. Il Capitolo 14 prende in esame la struttura di dati nota come array.

Array

Introduzione

In questo capitolo verrà introdotto il concetto di array. Gli array sono essenziali per la manipolazione di insiemi di variabili correlate. PHP supporta sia gli array a una dimensione sia quelli multidimensionali. Gli array possono essere utilizzati per raccogliere dati, come nomi di persone, ed eseguire operazioni su questi dati nel modo più semplice possibile. Inizieremo illustrando cosa si intende con array a una dimensione e come crearne uno.

Array a una dimensione

Gli array a una dimensione sono contenitori in grado di memorizzare una singola sequenza di elementi indicizzati. Gli array sono simili alle variabili, ma mentre queste ultime possono memorizzare un solo valore, gli array possono contenerne diversi. La Figura 14.1 illustra un array a una dimensione contenente un elenco di ortaggi.

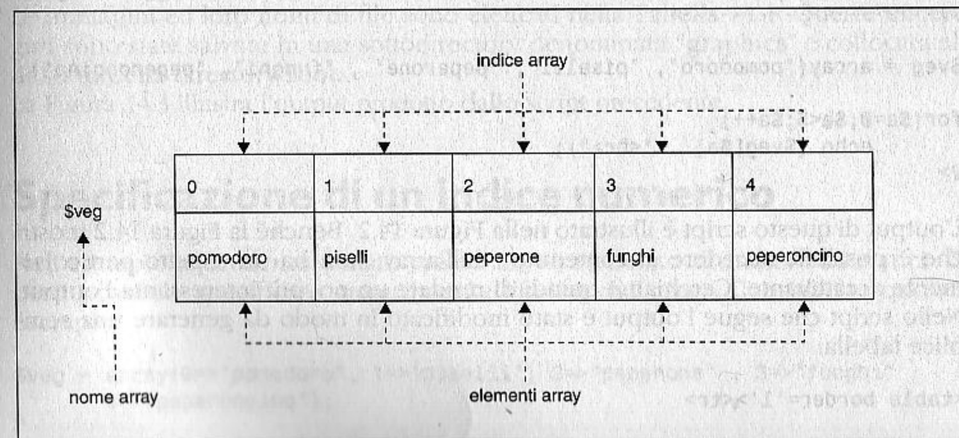


Figura 14.1
Array a una dimensione.

L'array è caratterizzato da un nome, **$veg**, e da diversi elementi, ciascuno dei quali contiene un valore, che in questo caso è un ortaggio. Ogni elemento ha un indice, che di default viene creato automaticamente a partire dal numero 0 e incrementato di 1 per ciascun elemento dell'array.

Costrutto array

In PHP la creazione degli array avviene con il costrutto `array`, che ha la forma seguente:

```
array = array([mixed...])
```

Per creare l'array illustrato nella Figura 14.1 è possibile dichiararlo nel modo seguente:

```
$veg = array("pomodoro", "piselli", "peperone" , "funghi", "peperoncino";
```

Questa istruzione crea un array chiamato **$veg** contenente cinque elementi che memorizzano ortaggi. Il riferimento agli elementi dell'array può avvenire aggiungendo un indice al nome dell'array. Dato che in base alle regole predefinite la numerazione degli indici di questo array va da 0 a 4, l'accesso al primo elemento dell'array può avvenire con l'istruzione seguente:

```
echo($veg[0]);
```

mentre questa istruzione riguarda l'ultimo elemento:

```
echo($veg[4]);
```

Lo script che segue dichiara l'array degli ortaggi e visualizza il contenuto dell'array con il ricorso a un ciclo `for`:

```
<?php
// Array - Esempio 14-1
//---------------------

$veg = array("pomodoro", "piselli", "peperone" , "funghi", "peperoncino");

for($a=0;$a<5;$a++)
      echo ($veg[$a] . "<br>");
?>
```

L'output di questo script è illustrato nella Figura 14.2. Benché la Figura 14.2 mostri che è possibile accedere al contenuto dell'array, non ha un aspetto particolarmente accattivante. Cerchiamo quindi di rendere un po' più interessante l'output. Nello script che segue l'output è stato modificato in modo da generare una semplice tabella:

```
<table border='1'><tr>

<?php
// Array - Esempio 14-2
//---------------------
```



```
$veg = array("pomodoro", "piselli", "peperone" , "funghi", "peperoncino");
for($a=0;$a<5;$a++)
      echo("<td><img src='graphics/" . $veg[$a] . ".jpg'></td>");
echo("</tr>");
?>
</table>
```

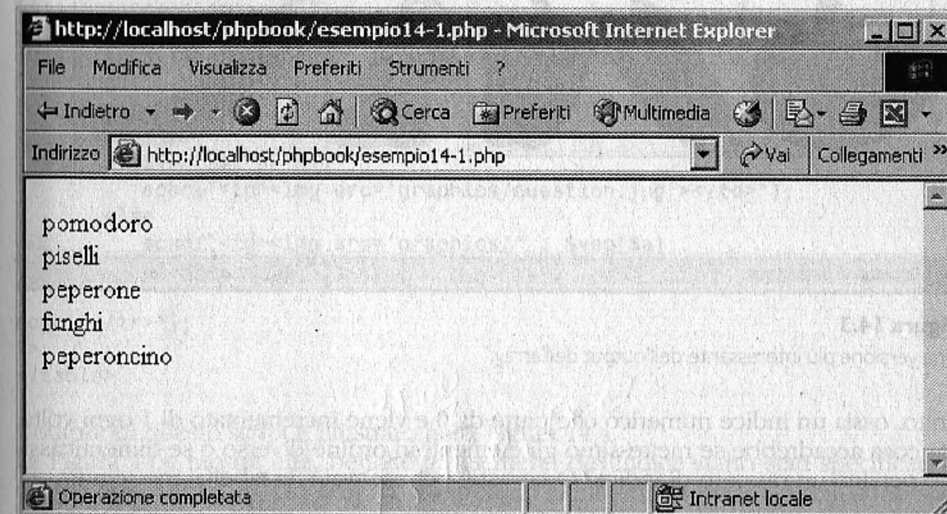


Figura 14.2
Visualizzazione di un array.

Lo script precedente visualizza una tabella le cui celle contengono i valori dell'array. Tuttavia, per rendere più interessante il risultato, sono state create semplici immagini che vengono utilizzate per rappresentare gli ortaggi contenuti nell'array; queste immagini sono state create e salvate con lo stesso nome con cui gli ortaggi sono memorizzati nell'array, con l'aggiunta dell'estensione .jpg. Le immagini e i loro nomi di file sono elencati nella Tabella 14.1. Queste immagini sono state salvate in una sottodirectory denominata "graphics" e collocata al di sotto della directory root.
La Figura 14.3 illustra l'output prodotto dallo script precedente.

Specificazione di un indice numerico

Nell'esempio precedente è stato creato un array dotato di indici automatici. È però possibile specificare manualmente l'indice dell'array. Considerate questo esempio:

```
$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone" , 3=>"funghi"
      4=>"peperoncino");
```

dove la specificazione di un indice può avvenire tramite l'operatore "=>". Nell'esempio precedente l'indice è stato specificato esattamente come quello prede-

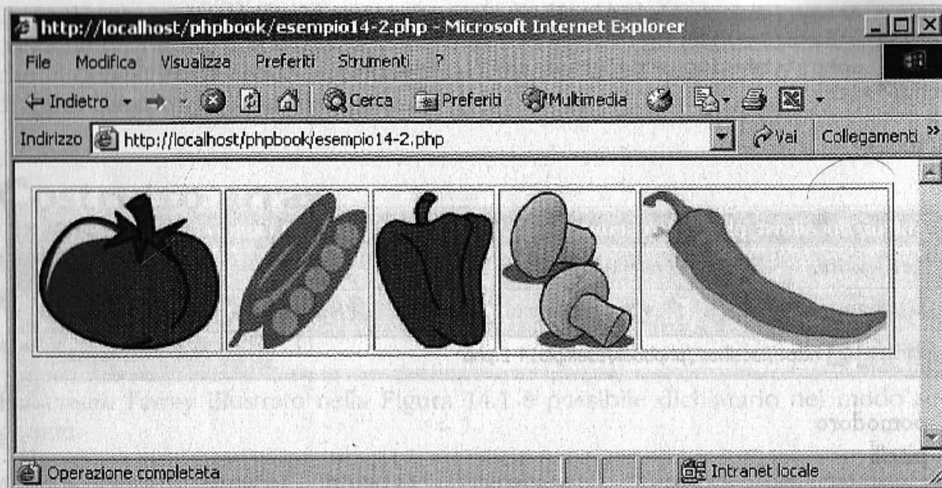


Figura 14.3
Una versione più interessante dell'output dell'array.

finito, ossia un indice numerico che parte da 0 e viene incrementato di 1 ogni volta. Ma cosa accadrebbe se mettessimo gli elementi in ordine diverso o se dimenticassimo persino un numero di indice? Osservate, per esempio, la tabella che segue.

Tabella 14.1 Immagini degli ortaggi

Immagine	Nome del file
	pomodoro.jpg
	peperone.jpg
	piselli.jpg
	funghi.jpg
	peperoncino.jpg

Nell'esempio precedente l'indice non viene fornito in ordine numerico. Per illustrare cosa accade in questo caso, l'array è stato incluso nello script seguente:

```
<table border='1'><tr>

<?php
//Array - Esempio 14-3
//---------------------

$veg = array(0=>"pomodoro", 2=>"piselli", 1=>"peperone", 3=>"",
4=>"funghi", 5=>"peperoncino");
for($a=0;$a<6;$a++)
{
    if($veg[$a] == "")
        echo("<td><img src='graphics/question.jpg'></td>");
    else
        echo("<td><img src='graphics/" . $veg[$a] .
        ".jpg'></td>");
}
echo("</tr>");
?>
</table>
```

L'output di questo script è illustrato nella Figura 14.4.
La Figura 14.4 mostra che, benché gli elementi dell'indice siano stati specificati in ordine non numerico, ciò non ha comportato alcuna differenza. I piselli sono stati collocati all'indice 2 e il peperone all'indice 1, ed è così che appaiono. Nel caso dell'elemento di indice 3, che è stato specificato come vuoto, non c'è alcun ortaggio da inserire nell'output, pertanto viene visualizzata l'immagine question.jpg. Questo avviene con l'aggiunta allo script di un'istruzione che controlla se l'indice è vuoto:

```
if ($veg[$a] == "")
      echo("<td><img src 'graphics/question.jpg'>< td>");
```

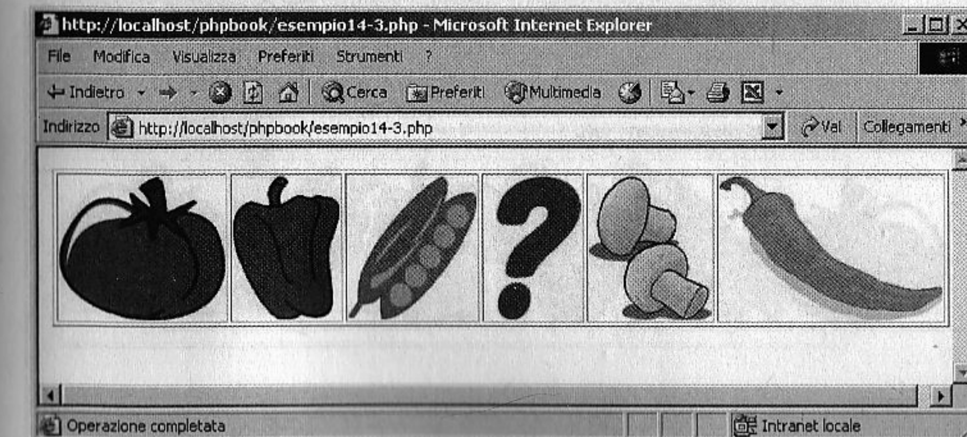


Figura 14.4
Specificazione dell'output con un proprio indice.

Ciclo foreach

PHP include un costrutto di ciclo progettato in modo specifico per l'iterazione attraverso gli array. Si tratta del ciclo `foreach`, che presenta due forme sintattiche. La prima è la seguente:

```
foreach ($array as $valore) istruzione
```

In questo caso il ciclo viene iterato attraverso l'array fornito da *`$array`*. Il valore dell'indice corrente viene assegnato a *`$valore`*. L'indice dell'array viene poi incrementato di uno in modo che la successiva iterazione del ciclo accederà all'elemento successivo. Lo script seguente presenta una riscrittura dell'esempio precedente al fine di includere questo ciclo:

```
<table border='1'><tr>
<?php

// Array - Esempio 14-4
//---------------------

$veg = array(0=>"pomodoro", 2=>"piselli", 1=>"peperone",
      4=>"funghi", 5=>"peperoncino");
foreach($veg as $vegItem)
      echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr>");
?>
</table>
```

Questo script è meno complesso rispetto all'esempio precedente. Osservate inoltre che non è più necessario controllare se ci sono posizioni vuote, in quanto il ciclo `foreach` si porta automaticamente al successivo elemento indicizzato, come illustra l'output della Figura 14.5.

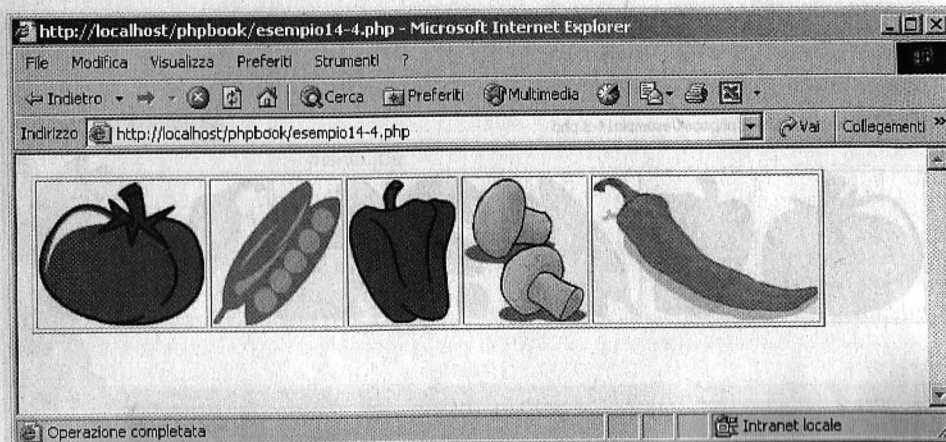


Figura 14.5
Output del ciclo foreach.

Osservate poi che l'output degli ortaggi avviene nell'ordine in cui vengono dichiarati, non in base all'indice numerico, come accade con il ciclo `for`.
La seconda forma del ciclo `foreach` è la seguente:

```
foreach ($array as $chiave => $valore) istruzione
```

Questa istruzione `foreach` in pratica funziona come l'esempio precedente, ma in più assegna il valore dell'indice dell'elemento corrente alla variabile *`$chiave`*, come mostrato nello script seguente:

```
<table border='1'><tr>

<?php

//Array - Esempio 14-5
//--------------------

$veg = array(0=>"pomodoro", 2=>"piselli", 1=>"peperone", 4=>"funghi",
      5=>"peperoncino");
foreach($veg as $vegItem)
      echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr><tr>");
foreach($veg as $key=>$vegItem)
      echo("<td align='center'>$key</td>");
?>
</tr></table>
```

Questo script utilizza due cicli `foreach`: il primo produce le immagini che rappresentano gli ortaggi contenuti nell'array `$veg`; il secondo invece accede alla chiave dell'indice di ciascun elemento dell'array e visualizza il valore relativo in una seconda riga della tabella sotto le immagini degli ortaggi. L'output di questo script è illustrato nella Figura 14.6.

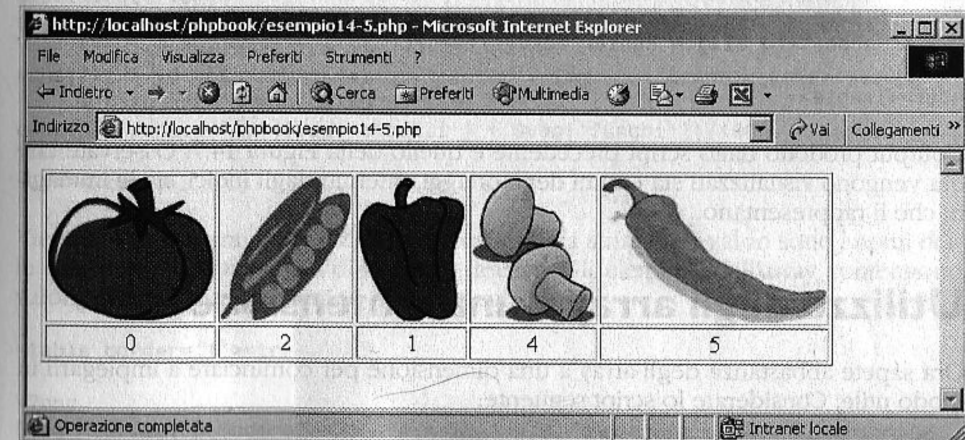


Figura 14.6
Output degli indici con foreach.

Ma per quale motivo si dovrebbe accedere ai valori di indice? Si è già visto come calcolarli utilizzando gli esempi di ciclo **foreach** presentati in precedenza in questo capitolo. Il fatto è che i valori di indice non devono necessariamente essere numerici: si vedrà che cosa significhi tutto ciò nella pratica, specificando alcuni indici non numerici.

Specificazione di un indice non numerico

Invece di specificare un array con indice numerico, è possibile specificarlo con un indice stringa. Per esempio:

```
$veg = array("Pomodoro"=>"pomodoro", "Piselli"=>"piselli",
        Peperone"=>"peperone", "Funghi"=>"funghi"
        "Peperoncino"=>"peperoncino");
```

L'istruzione precedente dichiara ancora una volta un array costituito da ortaggi; tuttavia, invece di utilizzare un indice numerico, specifica un indice sotto forma di stringhe che rappresentano i valori memorizzati negli elementi dell'array. Lo script seguente illustra l'uso di questa tipo di array:

```
<table border-'1'><tr>

<?php

// Array - Esempio 14-6
//--------------------

$veg = array("Pomodoro"=>"pomodoro", "Piselli"=>"piselli",
        "Peperone"=>"peperone", "Funghi"=>"funghi",
        "Peperoncino"=>"peperoncino");
foreach($veg as $vegItem)
        echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr><tr>");
foreach($veg as $key=>$vegItem)
        echo("<td align='center'>$key</td>");
?>
</tr></table>
```

L'output prodotto dallo script precedente è quello della Figura 14.7. Osservate che ora vengono visualizzati sia i nomi degli ortaggi, ottenuti dagli indici, sia le immagini che li rappresentano.

Utilizzo degli array a una dimensione

Ora sapete abbastanza degli array a una dimensione per cominciare a impiegarli in modo utile. Considerate lo script seguente:

```
<table border='1'><tr>

<?php

// Array - Esempio 14-7
//--------------------
```

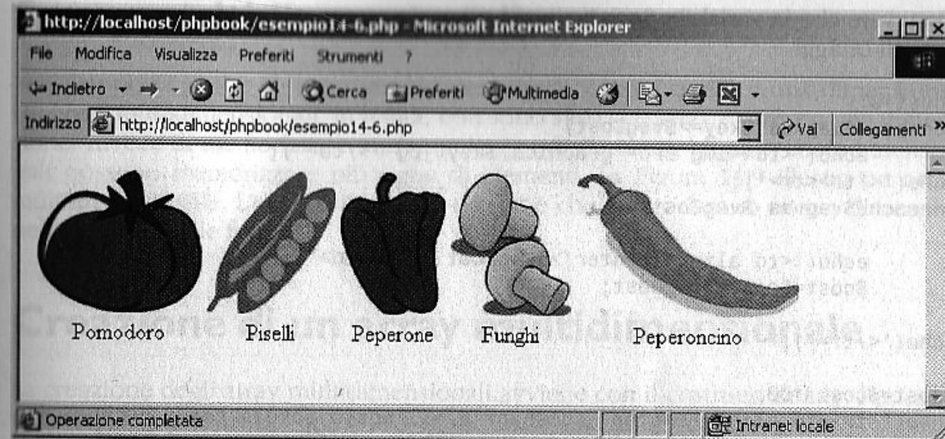



Figura 14.7
Output dell'array con indici stringa.

```
$veg = array("pomodoro"=>23, "piselli"=>78, "peperone"=>34, "funghi"=>56,
        "peperoncino"=>97);

$cost=0;
foreach($veg as $key=>$vegCost)
        echo("<td><img src='graphics/$key.jpg'></td>");
echo("</tr><tr>");
foreach($veg as $vegCost)
{
        echo("<td align='center'>$vegCost cent</td>");
        $cost=$cost+$vegCost;
}
echo("</tr>");
$cost=$cost/100;
echo("<tr><td align='center' colspan='5'>Il costo totale degli ortaggi è
     $cost euro</td></tr>");
echo("<tr><td colspan='5' align='center'><img src='graphics/piselli.jpg'>
     + <img src='graphics/funghi.jpg'>");
echo(" è pari a " . ($veg['piselli'] + $veg['funghi'])/100 . " euro.");
?>
</tr></table>
```

Viene creato un array i cui indici sono costituiti da stringhe. I valori sono i nomi delle immagini, e consentono di accedere a esse. Gli elementi dell'array contengono valori interi che rappresentano il costo degli ortaggi:

```
<table border='1'><tr>

<?php
$veg = array("pomodoro"=>23, "piselli"=>78, "peperone"=>34, "funghi"=>56
        "peperoncino"=>97);
```

Vengono utilizzati due cicli **foreach** per visualizzare le immagini degli ortaggi e i loro costi.

Osservate che la variabile **$cost** viene utilizzata per memorizzare il costo totale di tutti gli ortaggi:

```
$cost=0;
foreach($veg as $key=>$vegCost)
        echo("<td><img src='graphics/$key.jpg'></td>");
echo("</tr><tr>");
foreach($veg as $vegCost)
{
        echo("<td align='center'>$vegCost cent</td>");
        $cost=$cost+$vegCost;
}
echo("</tr>");

$cost=$cost/100;
echo("<tr><td align='center' colspan='5'>Il costo totale degli ortaggi è
$cost euro</td></tr>");
```

La parte finale dello script illustra quale sia la forza degli array dotati di indici stringa. In questo esempio viene sommato e visualizzato il valore di piselli e funghi. Il codice risulta molto più leggibile poiché vengono utilizzati i valori "piselli" e "funghi" invece che valori numerici.

```
echo("<tr><td colspan='5' align='center'><img src='graphics/piselli.jpg'>
      + <img src='graphics/funghi.jpg'>");
echo(" è pari a " . ($veg['piselli'] + $veg['funghi'])/100 . " euro.");
?>
</tr></table>
```

La Figura 14.8 illustra l'output prodotto dallo script precedente.

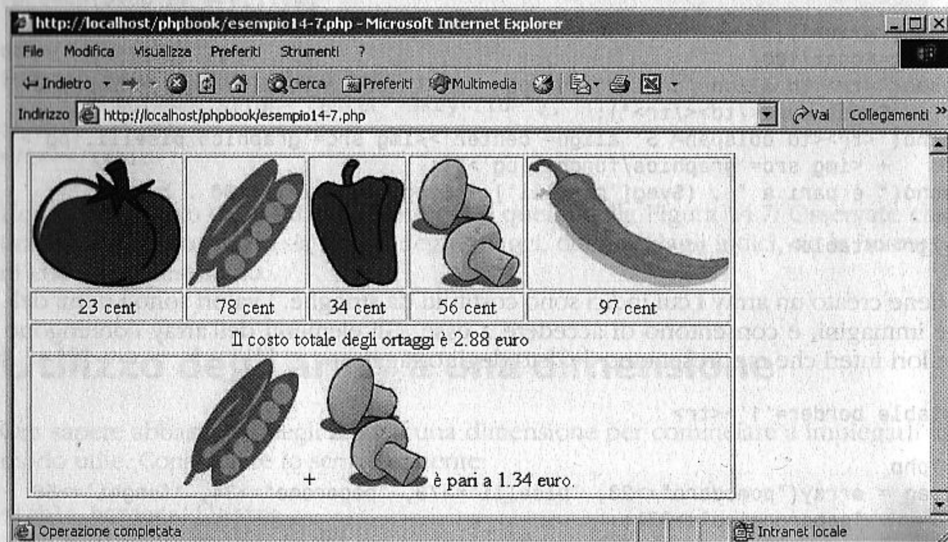


Figura 14.8
Utilizzo degli array.

Array multidimensionali

PHP supporta anche gli array multidimensionali, oltre a quelli a una dimensione. Gli array multidimensionali, in realtà, non sono altro che array di array a una dimensione. Invece di memorizzare una singola riga di elementi, gli array multidimensionali possono memorizzare più righe di elementi. La Figura 14.9 illustra un array multidimensionale. L'array è costituito da righe che memorizzano nome, dimensione e numero delle forme.

Creazione di un array multidimensionale

La creazione degli array multidimensionali avviene con il costrutto **array**. Per creare l'array illustrato nella Figura 14.9 si utilizza la sintassi seguente:

```
$shapes = array(
            array("stella", "sole", "ruota", "fiore", "macchia"),
            array(50,100,25,125,150),
            array(8,4,14,3,2)
);
```

Il costrutto è costituito dall'array **$shapes** che contiene tre array a una dimensione senza nome, ciascuno dei quali contiene separatamente il nome, la dimensione e la quantità di forme.

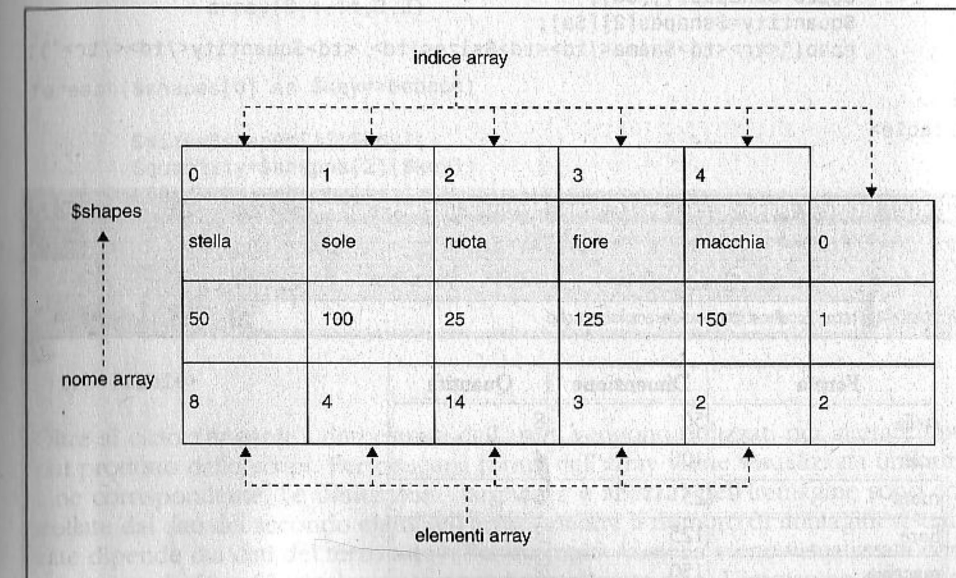


Figura 14.9
Array multidimensionale.

L'accesso ai diversi array può avvenire con un valore di indice, come avviene con gli array a una dimensione; poiché questo array ha due dimensioni, è necessario utilizzare due indici. Per esempio:

```
$var = $shapes[0][2];
```

Questa istruzione accede all'elemento 2 dell'array 0 (il primo array), che in questo esempio è "ruota". Lo script che segue illustra l'accesso ai dati contenuti nell'array attraverso un ciclo `for`:

```
<table border='1'>
<tr><th width=150>Forma</th><th width=100>Dimensione</th><th
width=100>Quantità</th></tr>

<?php

//Array - Esempio 14-8
//---------------------

$shapes = array(
            array("stella","sole","ruota","fiore","macchia"),
            array(50,100,25,125,150),
            array(8,4,14,3,2)
);
for($a=0;$a<5;$a++)
{
      $name=$shapes[0][$a];
      $size=$shapes[1][$a];
      $quantity=$shapes[2][$a];
      echo("<tr><td>$name</td><td>$size</td> <td>$quantity</td></tr>");
}
?>
</table>
```

http://localhost/phpbook/esempio14-8.php - Microsoft Internet Explorer

File Modifica Visualizza Preferiti Strumenti ?

Indietro Cerca Preferiti Multimedia

Indirizzo http://localhost/phpbook/esempio14-8.php Vai Collegamenti

Forma	Dimensione	Quantità
stella	50	8
sole	100	4
ruota	25	14
fiore	125	3
macchia	150	2

Operazione completata Intranet locale

Figura 14.10
Output dell'array multidimensionale.

Il programma precedente utilizza un singolo ciclo `for` per accedere al contenuto dell'array. Osservate che, dato che l'array `$shapes` è costituito da altri tre array, sono stati utilizzati gli indici `[0]`, `[1]` e `[2]` per accedere a tutti gli elementi di ciascuno dei tre array. L'output prodotto dallo script precedente è illustrato nella Figura 14.10.

Utilizzo di foreach con un array multidimensionale

Come per gli array a una dimensione, è possibile utilizzare il ciclo `foreach` per accedere al contenuto dell'array. Rivediamo lo script dell'esempio precedente per produrre un output interessante. Lo script che segue utilizza lo stesso array di prima, ma l'accesso ai dati avviene con un ciclo `foreach`:

```
<table border='1'>
<tr>

<?php

//Array - Esempio 14-9
//---------------------

$shapes = array(
            array("stella","sole","ruota","fiore","macchia"),
            array(50,100,25,125,150),
            array(8,4,14,3,2)
);

foreach($shapes[0] as $key=>$shape)
{
      $size=$shapes[1][$key];
      $quantity=$shapes[2][$key];
      echo("<td valign='top'>");
      for($count=0;$count<$quantity;$count++)
            echo("<img src='graphics/$shape.jpg' width='$size'
height='$size'><br>");
      echo("</td>");
}
?>
</tr></table>
```

Oltre al ciclo `foreach`, i dati estratti dall'array vengono utilizzati per alterare l'output prodotto dallo script. Per ciascuna forma dell'array viene visualizzata un'immagine corrispondente. Le dimensioni (larghezza e altezza) dell'immagine sono controllate dai dati del secondo elemento array, mentre il numero di immagini visualizzate dipende dai dati del terzo array. Per esempio, la stella viene visualizzata come immagine da 50 × 50 pixel e ne vengono visualizzate otto. L'immagine della ruota invece ha dimensioni 25 × 25 pixel e ne vengono visualizzate 14. L'output prodotto dallo script precedente è illustrato nella Figura 14.11.

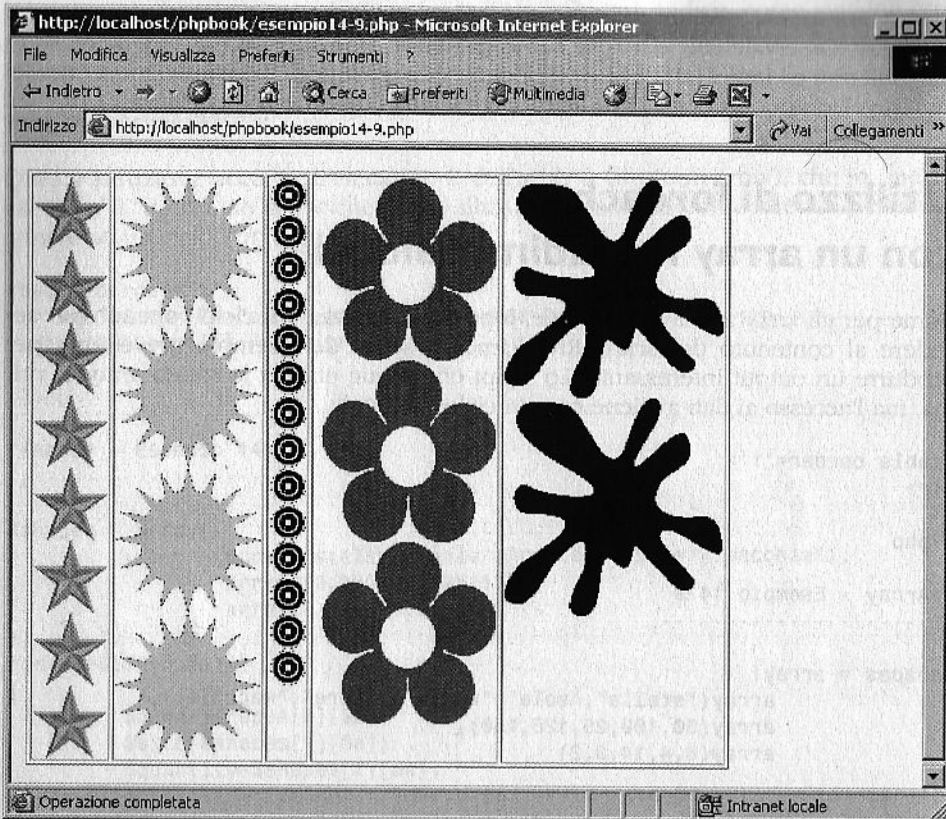


Figura 14.11
Output dell'array multidimensionale.

Utilizzo di un indice non numerico con un array multidimensionale

Nei nostri esempi precedenti di array multidimensionali non è stato specificato alcun indice, quindi l'array aveva un indice numerico intero predefinito. È tuttavia possibile specificare un proprio indice numerico o persino utilizzare delle stringhe. Per esempio:

```
$shapes = array(
            'immagine'=>
array("stella","sole","ruota","fiore","macchia"),
            'dimensione'=> array(50,100,25,125,150),
            'quantità'=> array(8,4,14,3,2)
);
```


Nell'esempio precedente ciascuno dei tre array ha un valore di indice: `immagine`, `dimensione` e `quantità`. Quella che segue è una riscrittura dello script precedente per illustrare come si possono utilizzare questi indici:

```
<table border='1'>
<tr>

<?php

//Array - Esempio 14-10
//--------------------

$shapes = array(
                'immagine'=>
array("stella","sole","ruota","fiore","macchia"),
                'dimensione'=> array(50,100,25,125,150),
                'quantità'=> array(8,4,14,3,2)
);
foreach($shapes['immagine'] as $key=>$shape){
            $size=$shapes['dimensione'][$key];
            $quantity=$shapes['quantità'][$key];
            echo("<td valign='top'>");
            for($count=0;$count<$quantity;$count++)
                echo("<img src='graphics/$shape.jpg' width='$size'
height='$size'><br>");
            echo("</td>");
}
?>
</tr></table>
```

L'output di questo script è lo stesso della Figura 14.11. È anche possibile specificare i valori di indice per ciascun elemento dell'array, come si è visto per l'array a una dimensione. Quello che segue è un esempio di quanto appena detto:

```
$shapes = array(
        'immagine'=>
      array(1=>"stella",2=>"sole",3=>"ruota",4=>"fiore",5=>"macchia"),
            'dimensione'=> array(1=>50,2=>100,3=>25,4=>125,5=>150),
            'quantità'=> array(1=>8,2=>4,3=>14,4=>3,5=>2)
);
```

Il codice precedente specifica che gli indici degli elementi degli array iniziano dal valore numerico 1 invece che da 0.

Riepilogo

Questo capitolo ha introdotto il concetto di array, mostrando come creare array a una dimensione e multidimensionali e come visualizzarne il contenuto. Nel prossimo capitolo si vedrà come si può accedere alla data e all'ora del sistema.

Capitolo 15

Date, ore e numeri casuali

Introduzione

La capacità di accedere all'ora e alla data di sistema è molto utile. Date e ore possono essere utilizzate per diverse operazioni, dalla semplice visualizzazione della data e dell'ora corrette sulla pagina Web alla creazione di un'indicazione del momento in cui un record di database è stato creato. Inoltre, si vedrà che la capacità di evidenziare l'ora è essenziale per consentire la creazione di numeri casuali. Questo capitolo introdurrà alcune funzioni chiave coinvolte nell'accesso alla data e all'ora e illustrerà com'è possibile creare numeri casuali.

Ottenere data e ora

L'accesso alla data e all'ora può avvenire attraverso una singola funzione chiamata `getdate()`.
La sintassi della funzione è la seguente.

```
array getdate();
```

La funzione `getdate()` restituisce un array contenente la data e l'ora corrente. Questo array è indicizzato tramite stringhe; i valori di indice e il contenuto sono illustrati nella Tabella 15.1.

Tabella 15.1 L'array getdate()

Nome	Descrizione
`seconds`	La parte dei secondi dell'ora corrente.
`minutes`	La parte dei minuti dell'ora corrente.
`hours`	La parte delle ore dell'ora corrente.
`mday`	Giorno del mese.
`wday`	Giorno della settimana in formato numerico (domenica = 0).
`mon`	Mese numerico.
`year`	Anno numerico.
`yday`	Giorno dell'anno in formato numerico, per esempio 312.
`weekday`	Giorno della settimana in formato testuale, ossia "Monday".
`month`	Mese dell'anno in formato testuale, ossia "May".

Segue un esempio della funzione:

```
$date = getdate();
```

Lo script che segue illustra l'accesso agli elementi dell'array della funzione `getdate()`:

```
<?php

// Date, ore e numeri casuali - Esempio 15-1
//--------------------------------

$date = getdate();

$seconds = $date['seconds'];
$minutes = $date['minutes'];
$hours = $date['hours'];
$mday = $date['mday'];
$wday = $date['wday'];
$mon = $date['mon'];
$year = $date['year'];
$yday = $date['yday'];
$weekday = $date['weekday'];
$month = $date['month'];

echo("Sono le ore $hours:$minutes:$seconds<br>");
echo("Oggi è il $mday/$mon/$year<br>");
echo("Siamo nel mese di $month e il giorno della settimana è $weekday<br>");
echo("È il giorno $wday di 6 di questa settimana (dove domenica è 0)<br>");
echo("Finora sono trascorsi $yday giorni di quest'anno.");

?>
```

Lo script richiama la funzione `getdate()` e assegna ciascun elemento dell'array restituito a variabili separate per una maggiore chiarezza. Queste vengono poi visualizzate sulla pagina Web, come illustrato nella Figura 15.1.

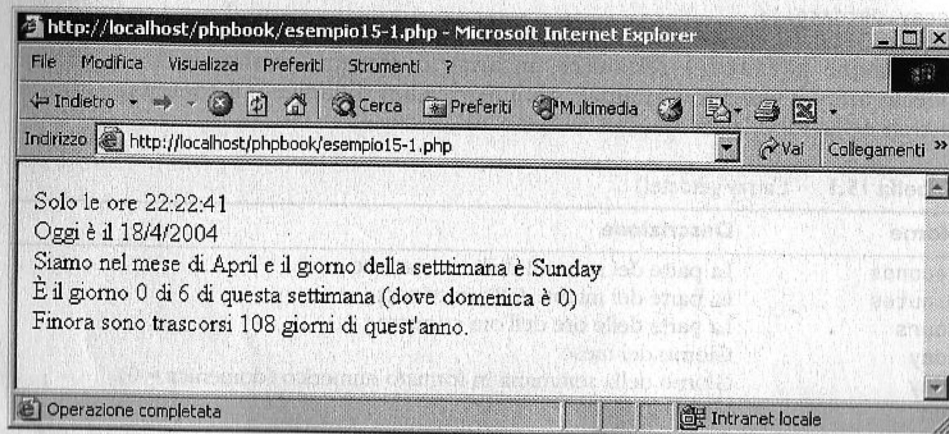


Figura 15.1
Valori di data e ora.

Ovviamente la data e l'ora visualizzata sul vostro computer saranno diverse da quelle della figura, poiché indicheranno quando eseguirete lo script. Al momento questi dati non sono utilizzati in alcun modo, ma solo visualizzati; tuttavia più avanti in questo capitolo e nei capitoli successivi ne vedrete qualche utile applicazione.

microtime

La funzione `getdate()` restituisce l'ora corrente al secondo più vicino. Benché si tratti di una funzione utile, potreste aver bisogno di un'indicazione oraria più precisa; questo è possibile con la funzione `microtime()`.
La sintassi della funzione è la seguente.

```
string microtime(void)
```

La tabella seguente descrive il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Restituzione di `microtime()`	`String`	Questa funzione restituisce una stringa costituita da due parti: "`msec sec`". Il valore di `sec` è il numero di secondi trascorsi dalla mezzanotte del 1 gennaio 1970. La parte `msec` della stringa è la frazione di microsecondi del secondo non intero corrente.

Esempio di funzione:

```
$secs = microtime();
```

Lo script seguente illustra un esempio di utilizzo di questa funzione:

```
<?php
// Date, ore e numeri casuali - Esempio 15-2
//--------------------------------

$secs = microtime();
echo($secs);
?>
```

Lo script precedente richiama la funzione `microtime()` e ne visualizza il contenuto. Benché al momento la funzione `microtime()` non mostri tutta la propria utilità, presto essa sarà evidente. Ritornerete a questa funzione più avanti nel capitolo, quando sarà studiata la generazione di numeri casuali.

Verifica di una data

A volte potreste volere che un utente inserisca una data. Sfortunatamente alcune date sono valide, mentre altre non lo sono. Considerate le date seguenti:

31 settembre 1402;
30 aprile 1999;

Solo alcune di esse sono valide, tuttavia è necessario un metodo per verificare questa condizione in uno script. Fortunatamente esiste la funzione `checkdate()`, che restituisce TRUE se la data che le viene passata è valida e FALSE se non lo è.
La sintassi della funzione è la seguente.

```
bool checkdate(int mese, int giorno, int anno);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
mese	int	Il mese.
giorno	int	Il giorno.
anno	int	L'anno.
Restituzione di `checkdate()`	bool	TRUE se la data è valida, FALSE se non lo è.

Esempio di funzione:

```
$date = checkdate(2,29,1402);
```

La funzione riceve tre parametri interi, che rappresentano il mese, il giorno e l'anno della data. Lo script che segue illustra un esempio di utilizzo di questa funzione:

```
<?php

// Date, ore e numeri casuali - Esempio 15-3
//--------------------------------

$date = checkdate(2,29,1402);
if($date)
	echo("Questa è una data valida!");
else
	echo("Questa non è una data valida");

?>
```

Lo script passa la data del 29 febbraio 1402 alla funzione `checkdate()` e poi visualizza un messaggio che indica se la data è o meno valida. Potete provare a cambiare la data passata alla funzione `checkdate()` per vedere se le date indicate in precedenza sono valide.

Generazione di un numero casuale

La capacità di generare numeri casuali è uno strumento utile in qualunque linguaggio di programmazione; senza di essa, per esempio, i giochi diventerebbero molto prevedibili.
In PHP i passaggi che portano alla generazione di un numero casuale sono due. Il primo riguarda l'inizializzazione con un parametro `seme` del generatore di numeri casuali con la funzione `srand()`.
La sintassi della funzione è la seguente.

```
void srand(int seme);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
seme	int	Seme casuale per il generatore di numeri casuali.
Restituzione di `srand()`	void	Non restituisce alcunché.

Esempio di funzione:

```
srand(1);
```

La funzione `srand()` viene utilizzata per impostare il generatore di numeri casuali a una posizione casuale, prima di ottenere i numeri casuali necessari allo script. La mancanza di questa inizializzazione produrrà un insieme di numeri casuali prevedibile. Per inizializzare il generatore di numeri casuali occorre partire da un numero molto grande, che può essere ricavato utilizzando la funzione `microtime()`; essa permette di ottenere la parte in microsecondi del suo output e moltiplicarla per un numero molto alto. Ciò è possibile con la seguente istruzione:

```
srand((double) microtime() * 1000000);
```

Questa linea inizializza il generatore di numeri casuali con un numero casuale prodotto a partire dall'ora corrente ottenuta dalla funzione `microtime()`. Ora siete pronti per ottenere numeri casuali, e per questo utilizzerete la funzione `rand()`.
La sintassi della funzione è la seguente.

```
int rand([int inizio, int fine]);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
inizio	int	Facoltativo. Valore minimo del numero casuale da generare.
fine	int	Facoltativo. Valore massimo del numero casuale da generare.
Restituzione di `rand()`	int	Numero casuale.

Esempio di funzione:

```
rand(10,1000);
```

La funzione `rand()` può essere chiamata senza alcun parametro, nel qual caso restituisce un numero casuale i cui valori minimo e massimo sono al di fuori del vostro controllo. Potete tuttavia limitare l'intervallo dei numeri casuali prodotti includendo i parametri di inizio e fine. Per esempio:

```
rand(10,1000);
```

Questa riga produrrà numeri casuali compresi tra 10 e 1000. Lo script che segue illustra l'utilizzo dello strumento di generazione di numeri casuali:

```
<?php
// Date, ore e numeri casuali - Esempio 15-4
//--------------------------------
```

```
srand((double) microtime() * 1000000);
$randVal = rand(1,6);
echo("Numero casuale: $randVal");
?>
```

L'esempio precedente produce semplicemente un numero casuale compreso tra 1 e 6, certamente nulla di particolarmente entusiasmante; se pensiamo però che la generazione di numeri casuali compresi tra 1 e 6 avviene comunemente nel mondo reale quando si gioca a dadi, la cosa assume un aspetto diverso. Vediamo allora se è possibile vivacizzare questo esempio.

Esempio dei dadi

In questo paragrafo verrà creata una pagina Web che visualizza il risultato del lancio di sei dadi a sei facce. La prima operazione da compiere è la creazione di sei immagini di dadi, illustrate nella Figura 15.2.

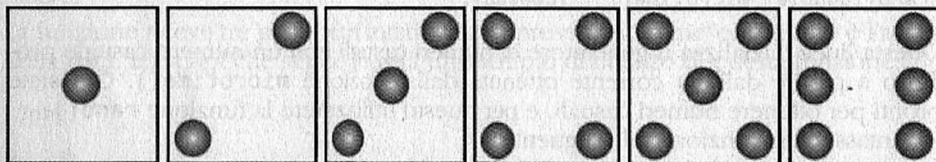


Figura 15.2
Immagini dei dadi.

Queste immagini sono memorizzate nella sottodirectory *graphics* con i nomi 1.jpg, 2.jpg e così via. Consideriamo ora uno script creato per utilizzarle:

```
<?php
// Date, ore e numeri casuali - Esempio 15-5
//-------------------------------

srand((double) microtime() * 1000000);
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
$randVal = rand(1,6);
echo("<img src='graphics/$randVal.jpg'>");
?>
```

Lo script precedente inizializza il generatore di numeri casuali e quindi ottiene numeri casuali compresi tra 1 e 6, che vengono memorizzati nella variabile `$randVal`. Questo valore viene poi inserito in un tag `<img>` per visualizzare la relativa immagine del dado. L'operazione viene ripetuta sei volte. L'output del programma precedente è illustrato nella Figura 15.3. Se fate clic sul pulsante *Aggiorna* del browser, i dadi visualizzati cambieranno in modo casuale.

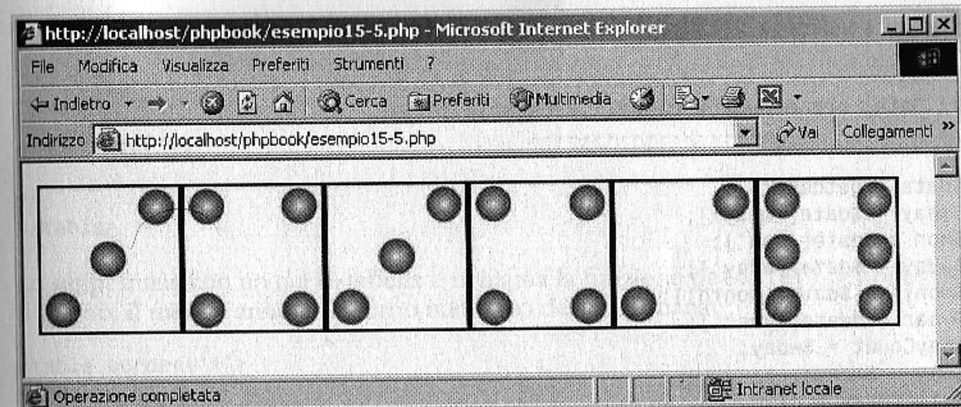


Figura 15.3
Esempio di lancio casuale di un dado.

Benché la Figura 15.3 mostri che è possibile vivacizzare lo script di generazione di numeri casuali, quest'ultimo non compie ancora alcuna operazione particolarmente utile. Torniamo quindi alle funzioni per le date e creiamo uno script che produrrà un utile calendario.

Implementazione di un calendario

La Figura 15.4 illustra un semplice calendario del mese di aprile.

April						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figura 15.4
Un semplice calendario.

A dispetto della sua apparente semplicità, questo calendario è in realtà piuttosto complesso. Per esempio, i giorni della settimana da domenica a sabato appaiono in colonne fisse, mentre il giorno di inizio del mese può essere uno qualunque di questi. Nell'esempio di aprile il primo giorno della settimana è giovedì, pertanto è ne

cessario visualizzare diverse celle vuote fino a collocare un 1 nella colonna del giovedì.
Per la creazione del calendario è necessario impiegare molte funzioni introdotte in questo capitolo:

```
<table border='1'>
<?php

// Date, ore e numeri casuali - Esempio 15-6
//----------------------------------

$date = getdate();
$mday = $date['mday'];
$mon = $date['mon'];
$wday = $date['wday'];
$month = $date['month'];
$year = $date['year'];
$dayCount = $wday;
$day = $mday;
while($day > 0)
{
        $days[$day--] = $dayCount--;
        if($dayCount < 0)
                $dayCount = 6;
}
$dayCount = $wday;
$day = $mday;

if(checkdate($mon,31,$year))
   $lastDay = 31;
elseif(checkdate($mon,30,$year))
   $lastDay = 30;
elseif(checkdate($mon,29,$year))
   $lastDay = 29;
elseif(checkdate($mon,28,$year))
   $lastDay = 28;
while($day <= $lastDay)
{
   $days[$day++] = $dayCount++;
   if($dayCount > 6)
      $dayCount = 0;
}

echo("<tr><td colspan='7' align='center'>$month</td></tr>");
echo("<tr><td>Sun</td><td>Mon</td><td>Tue</td><td>Wed</td><td>
     Thu</td><td>Fri</td><td>Sat</td></tr>");
$startDay = 0;
$d = $days[1];
echo("<tr>");
while($startDay < $d)
{
        echo("<td></td>");
        $startDay++;
}
for ($d=1;$d<=$lastDay;$d++){
        if($d == $mday)
```



```
                echo("<td bgcolor='lightblue'>$d</td>");
        else
                echo("<td>$d</td>");
        $startDay++;
        if($startDay > 6 && $d < $lastDay)
        {
                $startDay = 0;
                echo("</tr><tr>");
        }
}
echo("</tr>");

?>
</table>
```

Lo script inizia con un tag di tabella e richiama la funzione `getdate()` per ottene il giorno, il mese, l'anno e il giorno numerico della settimana:

```
<table border='1'>
<?php
$date = getdate();
$mday = $date['mday'];
$mon = $date['mon'];
$wday = $date['wday'];
$month = $date['month'];
$year = $date['year'];
```

Poi viene utilizzato un ciclo `while` per iniziare a popolare l'array `$days`, che cont ne il giorno della settimana numerico di ciascun giorno del mese. Questo cic `while` calcola il giorno della settimana per tutte le date a partire da quella corren sino all'inizio del mese:

```
$dayCount = $wday;
$day = $mday;
while($day > 0)
{
        $days[$day--] = $dayCount--;
        if($dayCount < 0)
                $dayCount = 6;
}
$dayCount = $wday;
$day = $mday;
```

Nell'esempio l'array `$days` appare effettivamente così, con il 18 come giorno mese corrente, che è una domenica:
A questo punto viene utilizzata la funzione `checkdate()` per determinare qua l'ultimo giorno del mese corrente, che potrebbe essere 31, 30, 29 o 28:

```
if(checkdate($mon,31,$year))
   $lastDay = 31;
elseif(checkdate($mon,30,$year))
   $lastDay = 30;
elseif(checkdate($mon,29,$year))
   $lastDay = 29;
elseif(checkdate($mon,28,$year))
   $lastDay = 28;
```

Il programma utilizza quindi un ciclo while per memorizzare nell'array $days il giorno della settimana per tutte le date del mese, a partire da quella corrente:

```
while($day <= $lastDay)
{
   $days[$day++] = $dayCount++;
   if($dayCount > 6)
      $dayCount = 0;
}
```

Nell'esempio l'array $days appare come indicato di seguito.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6

L'array precedente indica che, mentre il primo giorno del mese è giovedì, l'ultimo cade di sabato. Per creare il resto della tabella del calendario dovranno essere visualizzate innanzi tutto le intestazioni di colonna, con il mese e i giorni della settimana:

```
echo("<tr><td colspan='7' align='center'>$month</td></tr>");
echo("<tr><td>Sun</td><td>Mon</td><td>Tue</td><td>Wed</td><td>
     Thu</td><td>Fri</td><td>Sat</td></tr>");
```

Poi si ricava la data del primo giorno della settimana dall'array $days e la si memorizza nella variabile $d. Viene utilizzato un ciclo while per produrre una tabella vuota, sino ad arrivare alla colonna del giorno della settimana corretto:

```
$startDay = 0;
$d = $days[1];
echo("<tr>");
while($startDay < $d)
{
    echo("<td></td>");
    $startDay++;
}
```

A questo punto siete pronti a visualizzare tutti i giorni del mese, e per questo si ricorre a un ciclo for. Quando si raggiunge il giorno del mese corrente, il colore di sfondo della cella della tabella corrispondente è azzurro (come indica l'attributo lightblue) per evidenziarla:

```
for ($d=1;$d<=$lastDay;$d++){
    if($d == $mday)
           echo("<td bgcolor='lightblue'>$d</td>");
    else
           echo("<td>$d</td>");}
```

Il ciclo contiene una condizione if che verifica quando si raggiunge la fine della settimana, ossia un sabato.
Esso produce altresì un elemento di fine riga, per garantire la corretta formattazione della tabella:

```
    $startDay++;
    if($startDay > 6 && $d < $lastDay)
    {
           $startDay = 0;
           echo("</tr><tr>");
    }
}
echo("</tr>");

?>
</table>
```

L'output di questo script è illustrato nella Figura 15.5. Questo script potrebbe esse modificato, per esempio per consentire a un utente di fare clic su una data ch dev'essere immessa in un modulo.

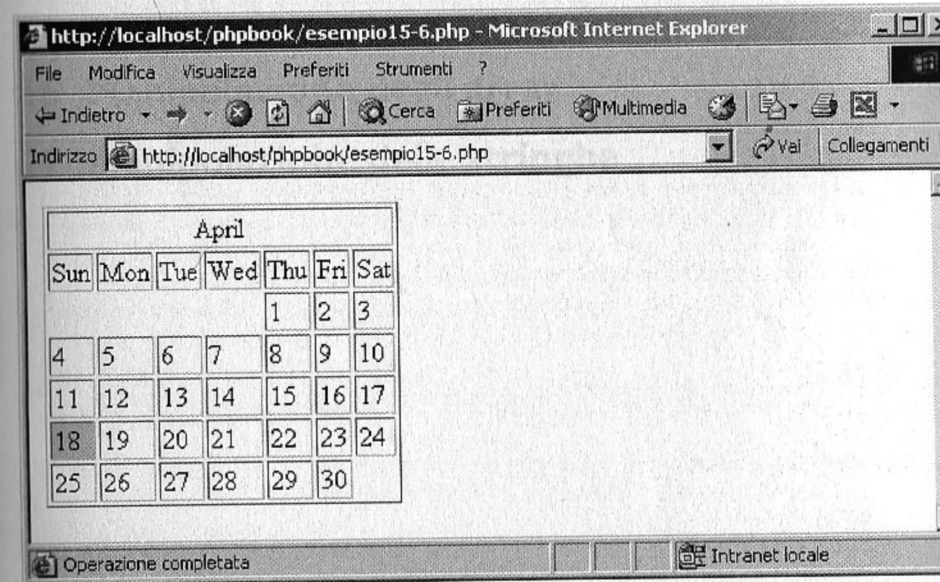


Figura 15.5
Output del calendario.

Riepilogo

Questo capitolo ha introdotto le funzioni che consentono di accedere alla data e di sistema. Si è visto come l'accesso all'ora sia molto importante per riuscire a ge rare numeri casuali. Il capitolo si è concluso con l'analisi della procedura atta a c are un calendario. Le funzioni di data e ora di questo capitolo hanno utilizzato ar per restituire i loro valori. Benché gli array siano stati introdotti nel capitolo pre dente, ora l'argomento verrà ripreso, soprattutto per considerare alcune funzioni sponibili per la loro manipolazione.

Manipolazione degli array

Introduzione

Nel Capitolo 14 è stato introdotto il concetto di array e fornito qualche esempio di come utilizzarli. Si è visto che PHP supporta sia i tipi di array a una dimensione sia quelli multidimensionali. Ma non è stato possibile esaurire l'argomento. Infatti, la libreria di PHP contiene molte funzioni che aiutano a manipolare agevolmente gli array. Questo capitolo presenterà alcune di queste funzioni e illustrerà come utilizzarle in modo che siano di aiuto nella programmazione.

Conteggio degli elementi di un array

In qualche caso potrebbe accadere che sia necessario contare il numero di elementi presenti in un array. Le funzioni `sizeof()` e `count()` offrono entrambe uno strumento utile allo scopo.
La sintassi delle funzioni è indicata di seguito.

```
int count(array Array);
int sizeof(array Array);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
Array	array	L'array da utilizzare.
Restituzione di `count()`	int	La dimensione dell'array.
Restituzione di `sizeof()`	int	La dimensione dell'array.

Esempi di funzioni:

```
$count = count($array);
$count = sizeof($array);
```

Lo script seguente illustra l'utilizzo della funzione `count()` per ottenere la dimensione dell'array, che viene poi utilizzata per controllare la terminazione del ciclo `for`:

```
<table border='1'><tr>
<?php

// Manipolazione degli array - Esempio 16-1
//-------------------------------------

$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi",
          4=>"peperoncino");

for($item=0;$item<count($veg);$item++)
          echo("<td><img src='graphics/" . $veg[$item] . ".jpg'></td>");
echo("</tr><tr><td align='center' colspan='". count($veg). "'>L'array
         contiene ". count($veg). "elementi.</td></tr>");

?>
</table>
```

Lo script visualizza anche un messaggio che indica il numero di elementi contenuti nell'array. Tale valore viene utilizzato per stabilire il numero di colonne su cui si dovrebbe impostare l'attributo `rowspan` dell'elemento `<TD>`, per assicurare che la prima cella della seconda riga della tabella occupi l'intera larghezza della tabella. In questo modo, il messaggio "L'array contiene..." sarà centrato nella tabella. L'output dello script precedente è illustrato nella Figura 16.1.

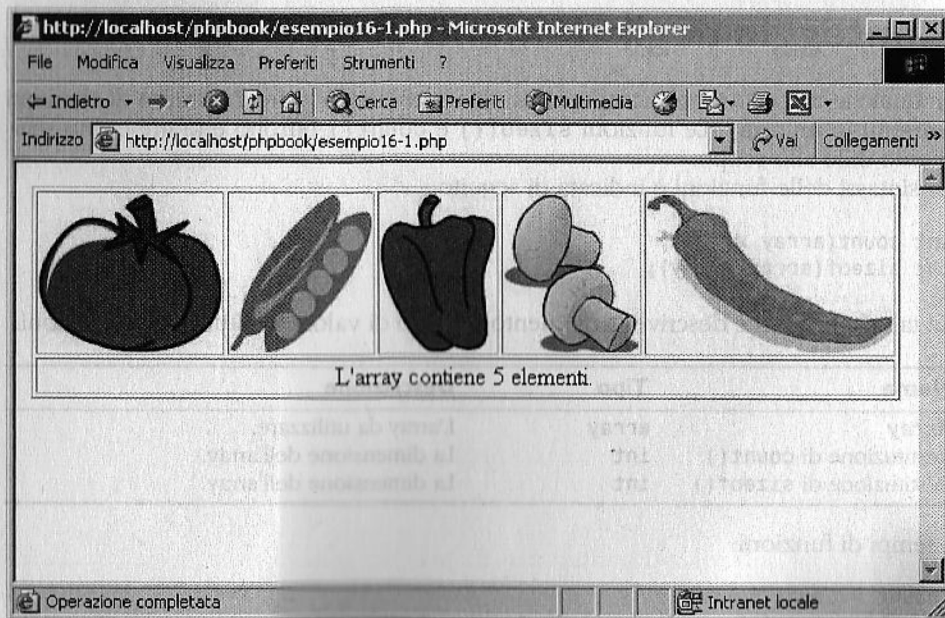


Figura 16.1
Conteggio degli elementi di un array.

Elementi casuali di un array

Dopo aver creato un array di elementi, potreste volerne randomizzare il contenuto, collocando ogni elemento dell'array in una posizione casuale. La funzione `shuffle()` consente di ottenere questo risultato in modo piuttosto agevole.
La sintassi della funzione è la seguente.

```
void shuffle(array Array);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Array	`array`	L'array da mescolare.
Restituzione di `shuffle()`	`void`	Non restituisce alcunché.

Esempio di funzione:

```
shuffle ($array) ;
```

Affinché la funzione `shuffle()` possa essere formattata correttamente, è necessario assegnare al generatore di numeri casuali un parametro `seme` con la funzione `srand()`, descritta nel Capitolo 15. Lo script che segue illustra l'utilizzo di questa funzione.

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-2
//-------------------------------------


srand((double)microtime()*1000000);

$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi",
          4=>"peperoncino");

for($count=0;$count<3;$count++){
          echo("<tr>");
          foreach($veg as $vegItem)
          echo("<td><img src='graphics/$vegItem.jpg'></td>");
          echo("</tr>");
          shuffle($veg);
}

?>
</table>
```

Lo script inizia assegnando un parametro `seme` al generatore di numeri casuali e prosegue con la creazione dell'array. Viene utilizzato un ciclo `for` con un ciclo `foreach` annidato per visualizzare tre volte il contenuto dell'array. Dopo ogni iterazione del ciclo `for` esterno, viene utilizzata una funzione `shuffle()` per rendere casuale il contenuto del ciclo. L'output di questo script visualizza il contenuto dell'array così come è stato creato e poi mostra il contenuto dell'array altre due volte dopo la randomizzazione. La Figura 16.2 illustra un tipico output di questo script.

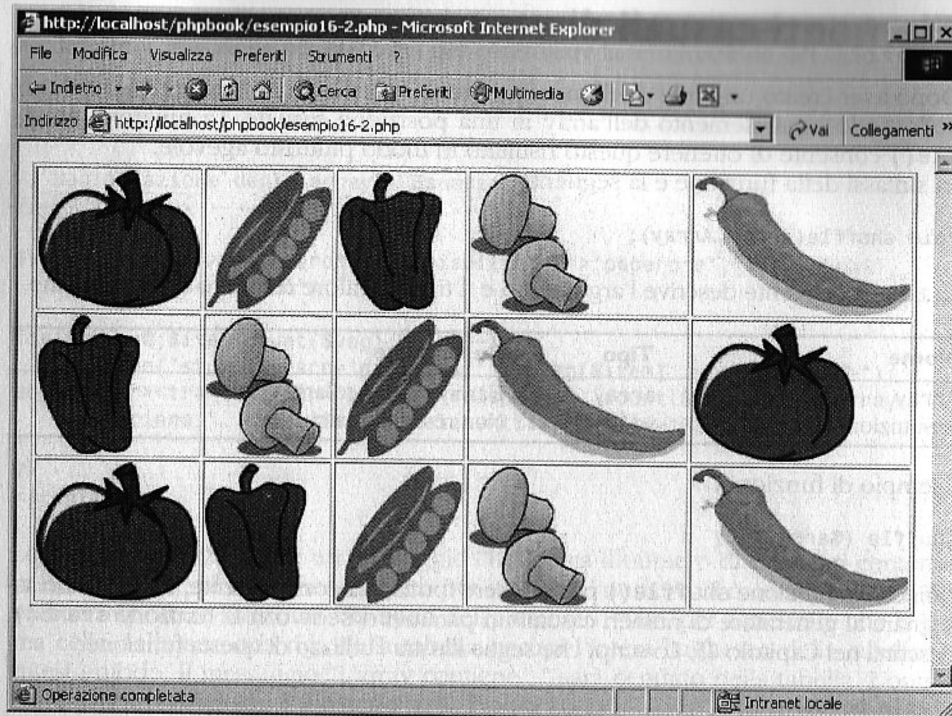


Figura 16.2
Randomizzazione di un array.

Selezione di elementi casuali

Invece di randomizzare l'intero array, si potrebbe scegliere di mantenerlo così come è stato creato e di estrarne semplicemente un elemento casuale. A tal fine è possibile utilizzare la funzione `array_and()`.
La sintassi della funzione è la seguente.

```
mixed array_rand(array Array);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Array	`array`	L'array da cui ottenere un elemento casuale.
Restituzione di `array_rand()`	`mixed`	Elemento casuale dell'array.

Esempio di funzione:

```
$pos = array_rand($veg);
```

La funzione `array_rand()` riceve un array sotto forma di parametro e restituisce l'indice di un elemento casuale dell'array. La funzione `array_rand()` richiede anche che il generatore di numeri casuali venga inizializzato con un parametro `seme`.

Lo script che segue mostra l'utilizzo della funzione:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-3
//-----------------------------------

srand((double)microtime()*1000000);
$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi",
        4=>"peperoncino");
for($rows=0;$rows<3;$rows++){
            echo("<tr>");
            for($cols=0;$cols<5;$cols++)
            {
                $pos = array_rand($veg);
                echo("<td><img src='graphics/" . $veg[$pos] . ".jpg'></td>");
            }
    echo("</tr>");
}

?>
</table>
```

Lo script inizia assegnando un parametro `seme` al generatore di numeri casuali e procede con la creazione di un array. Poi utilizza due cicli `for` annidati per visualizzare tre righe contenenti cinque elementi casuali dell'array. La Figura 16.3 illustra l'output dello script precedente.

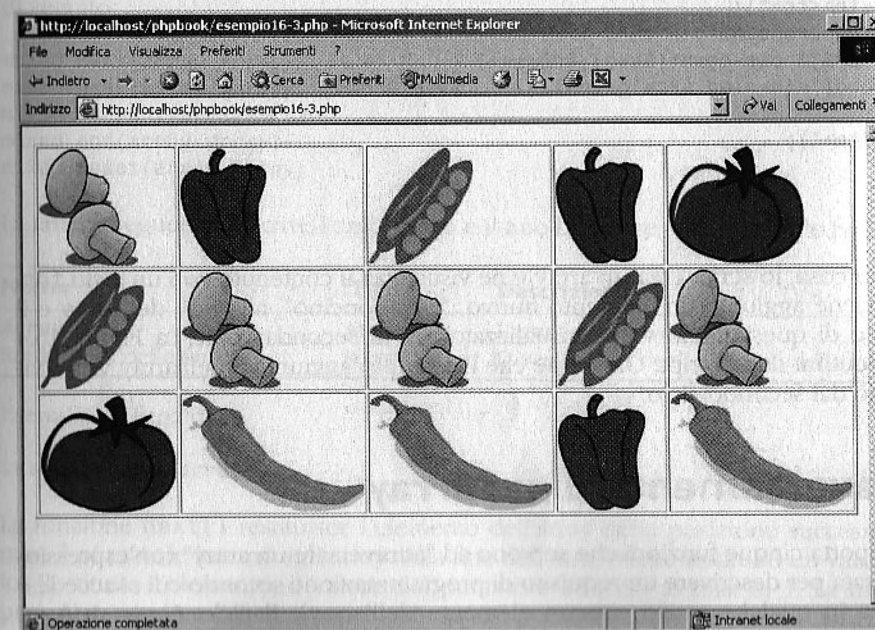


Figura 16.3
Estrazione di un elemento casuale da un array.

Mentre l'esempio precedente, come mostrato nella Figura 16.2, visualizza tutti gli elementi dell'array in ordine casuale, l'esempio nella Figura 16.3 visualizza elementi casuali, e in molti casi gli stessi ortaggi compaiono più volte.

Aggiunta di elementi alla fine di un array

Dopo aver creato un array, potreste voler aggiungere qualche elemento nuovo alla fine di esso. A questo scopo è possibile utilizzare il metodo seguente:

```
$array[] $newItem;
```

L'istruzione di assegnamento precedente attribuisce il valore `$newItem` all'array `$array`. Se non si specifica l'indice dell'array in cui inserire il valore, quest'ultimo sarà memorizzato automaticamente in un elemento nuovo alla fine dell'array. Lo script che segue illustra un esempio di questa operazione:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-4
//-----------------------------------


$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi");
echo("<tr>");
for($rows=0;$rows<count($veg);$rows++){
   echo("<td><img src='graphics/" . $veg[$rows] . ".jpg'></td>");
}
echo("</tr><tr>");
$veg[]="peperoncino";
for($rows=0;$rows<count($veg);$rows++) {
      echo("<td><img src='graphics/" . $veg[$rows] . ".jpg'></td>");
}
echo("</tr>");

?>
</table>
```

Per prima cosa, lo script crea un array e ne visualizza il contenuto con un ciclo `for`. Quindi viene aggiunto un elemento nuovo, "peperoncino", alla fine dell'array e il contenuto di quest'ultimo viene visualizzato per la seconda volta. La Figura 16.4 mostra l'output dello script. Osservate che l'elemento aggiuntivo dell'array viene visualizzato dal secondo ciclo.

Attraversamento di un array

PHP supporta cinque funzioni che servono ad "attraversare un array" (un'espressione utilizzata per descrivere un requisito di programmazione secondo cui si accede e si utilizza in qualche modo ciascun elemento dell'array). Il ciclo `foreach` è un esempio di costrutto che consente di attraversare un array. Questo ciclo, tuttavia, permette di attraversare l'array solo in un'unica direzione, dall'inizio alla fine. Le funzioni `reset()`, `next()`, `prev()`, `current()` ed `end()` consentono invece di

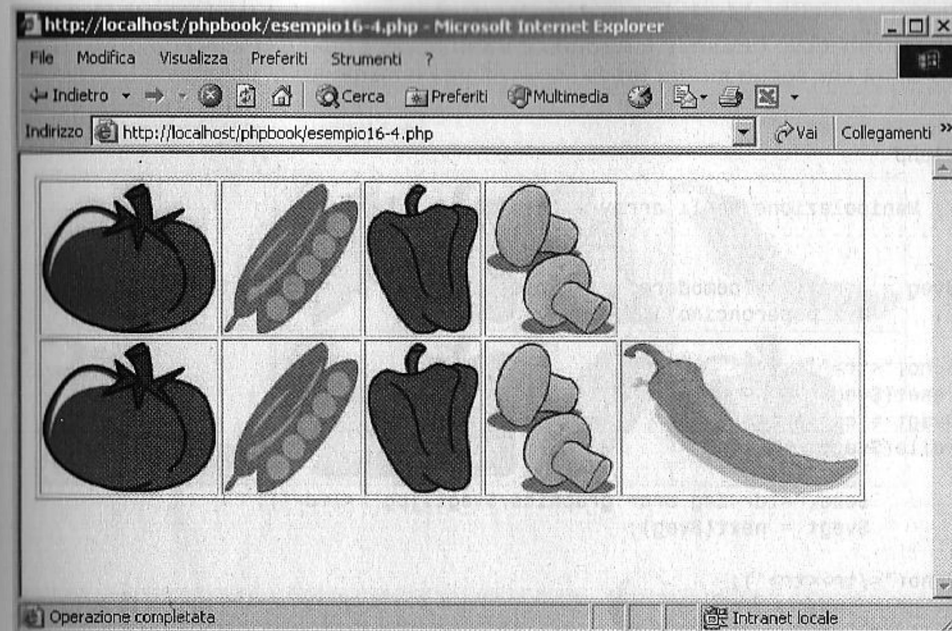


Figura 16.4
Aggiunta di un elemento alla fine di un array.

spostarsi facilmente da un elemento all'altro. La sintassi di queste funzioni è indicata di seguito.

```
mixed next(array Array)
mixed prev(array Array)
mixed current(array Array)
mixed end(array Array)
mixed reset(array Array)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
Array	`array`	L'array da attraversare.
Restituzioni delle funzioni	`mixed`	Diversi valori degli elementi dell'array.

Esempio di funzione:

```
$element end($array);
```

La funzione `next()` restituisce l'elemento dell'array nella posizione successiva cui mira il puntatore interno agli array. Alla fine dell'array viene restituito un valore nullo. Restituito l'elemento, il valore del puntatore all'array avanza di 1. La funzione `prev()` restituisce l'elemento precedente cui punta il puntatore interno all'array. Se si raggiunge l'inizio dell'array, viene restituito un valore 0. La funzione `current()` restituisce l'elemento dell'array cui attualmente si indirizza il puntatore e non sposta il puntatore interno. La funzione `end()` sposta il puntatore all'array interno alla fine

dell'array e restituisce l'ultimo elemento. La funzione `reset()` sposta il puntatore interno all'inizio dell'array e restituisce il valore del primo elemento.
Lo script che segue mostra un esempio di attraversamento di un array:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-5
//-----------------------------------

$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi",
       4=>"peperoncino");

echo("<tr>");
reset($veg);
$vegt = current($veg);
while($vegt)
{
       echo("<td><img src='graphics/$vegt.jpg'></td>");
       $vegt = next($veg);
}
echo("</tr><tr>");
$vegt = end($veg);
while($vegt)
{
       echo("<td><img src='graphics/$vegt.jpg'></td>");
       $vegt = prev($veg);
}
echo("</tr>");

?>
</table>
```

Nello script viene dichiarato un array, mentre il puntatore all'array viene ripristinato. Si ottiene l'elemento corrente dell'array (il primo). Con un ciclo `while` si visualizzano gli elementi dell'array e si ottiene l'elemento dell'array successivo. Alla fine dell'array, il puntatore viene impostato con la funzione `end()`. Un secondo ciclo `while` visualizza gli elementi dell'array e invoca la funzione `prev()` per attraversare nuovamente l'array. La Figura 16.5 mostra l'output dello script precedente. Come si può vedere, lo script è in grado di visualizzare il contenuto dell'array in ordine normale o inverso.

Inserimento ed estrazione: creazione di una pila

Gli array sono utilizzati per memorizzare le informazioni e, molto spesso, la quantità di informazioni aumenta o diminuisce. Accade di frequente di avere l'esigenza di implementare un array che permetta di aggiungere e rimuovere i dati alla fine dell'array. Tale operazione è conosciuta con il nome di "implementazione di una pila". L'aggiunta di dati alla fine della pila è nota come "inserimento", mentre la rimozione di dati prende il nome di "estrazione". PHP comprende due funzioni che consentono di inserire ed estrarre i dati.

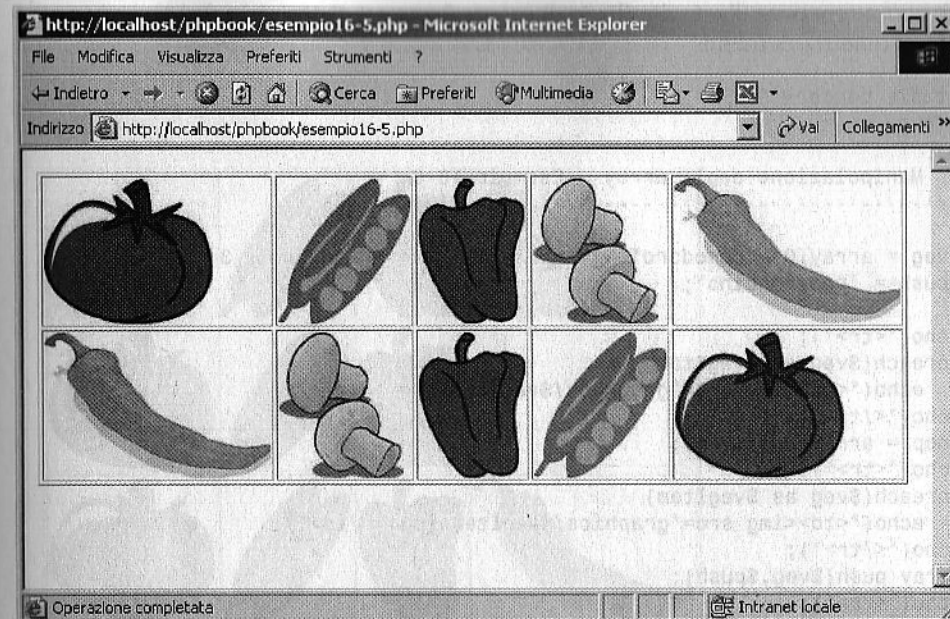


Figura 16.5
Attraversamento di un array.

La sintassi delle funzioni è indicata di seguito.

```
mixed array_pop(array Array)
int array_push(array Array,mixed Elemento_array)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
Array	`array`	L'array da utilizzare per l'estrazione/inserimento.
Elemento_array	`mixed`	Singolo elemento dell'array.
Restituzione di `array_pop()`	`mixed`	Valore memorizzato nell'ultimo elemento dell'array.
Restituzione di `array_push()`	`mixed`	Il numero di elementi memorizzati nell'array.

Esempi di funzioni:

```
$pop = array_pop($veg);
array_push($veg,$push);
```

La funzione `array_pop()` richiede un array come parametro, restituisce il valore memorizzato nell'ultimo elemento dell'array e riduce di 1 la dimensione dell'array. La funzione `array push()` richiede un array e il dato che si intende aggiungere all'array come parametro. La dimensione dell'array viene aumentata di un elemento e il dato viene memorizzato proprio in questo elemento. La funzione restituisce il numero di elementi memorizzati nell'array.

Lo script seguente illustra l'utilizzo di queste funzioni. Esso inizia dichiarando un array di quattro elementi:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-6
//-----------------------------------------

$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi");
$push = "peperoncino";

echo("<tr>");
foreach($veg as $vegItem)
   echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr>");
$pop = array_pop($veg);
echo("<tr>");
foreach($veg as $vegItem)
   echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr>");
array_push($veg,$push);
echo("<tr>");
foreach($veg as $vegItem)
   echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr>");

?>
</table>
```

Nello script viene creato un nuovo dato di nome `$push` che contiene la stringa "peperoncino", che sarà aggiunta successivamente all'array. Un ciclo `foreach` consente di visualizzare il contenuto dell'array, dopodiché viene invocata la funzione `array_pop()` per rimuovere l'ultimo elemento dall'array. Il contenuto dell'array viene nuovamente visualizzato per mostrare che l'ultimo elemento è stato rimosso. Infine viene chiamata la funzione `array_push()` per aggiungere la variabile `$push` alla fine dell'array. Il contenuto dell'array viene visualizzato ancora una volta per mostrare il nuovo contenuto dell'ultimo elemento. L'output creato dallo script precedente è quello della Figura 16.6.

Conteggio delle occorrenze

Ogni tanto gli array contengono alcuni elementi identici. Per esempio, supponete di chiedere a cento persone qual è il loro ortaggio preferito e di memorizzare le risposte nell'array. Una volta ottenuti i dati, dovete elaborarli per stabilire i risultati del sondaggio. La funzione `array_count_values()` è uno strumento molto utile per determinare il numero di elementi uguali contenuti in un array.
Il sintassi della funzione è la seguente.

```
array array_count_values(array Array)
```

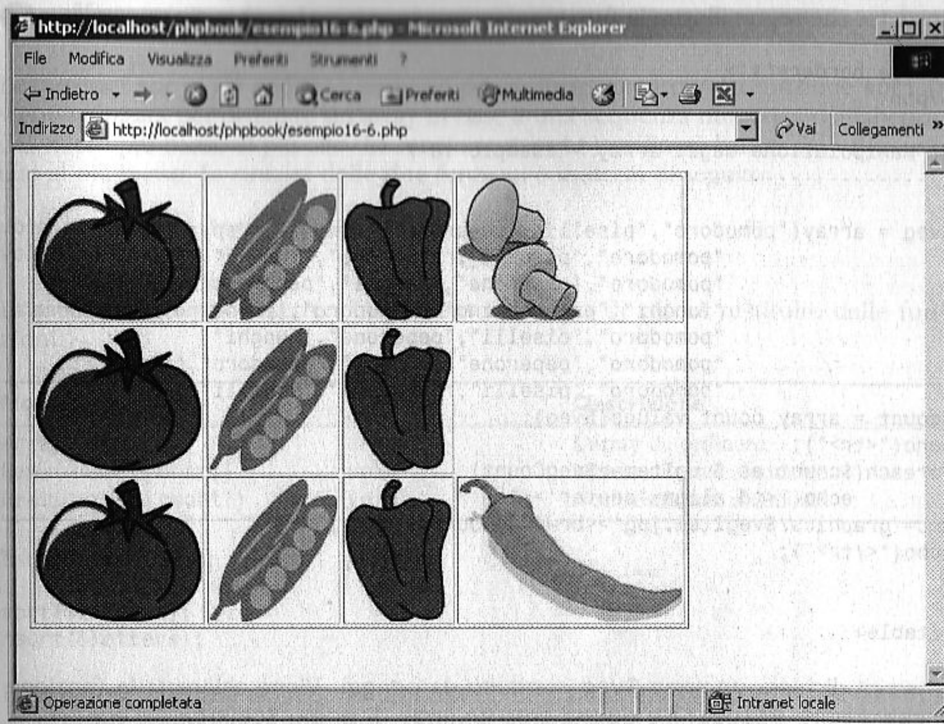



Figura 16.6
Inserimento ed estrazione.

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Array	array	L'array in cui contare gli elementi uguali.
Restituzione di `array_count_values()`	array	Restituisce un array di valori che indicano il numero di occorrenze di un particolare elemento.

Esempio di funzione:

```
$count = array_count_values($array);
```

La funzione richiede un array come parametro e restituisce un array dei risultati. Se per esempio, è stato creato un array costituito dai seguenti elementi:

```
$array = array("pomodoro","piselli","peperone","piselli","piselli");
```

invocando la funzione `array_count_values()` in questo modo:

```
$count = array_count_values($array);
```

si otterrà la creazione di un array `$count` con il contenuto seguente:

```
("pomodoro"=>1,"piselli"=>3,"peperone"=>1)
```

Lo script che segue illustra la funzione `array_count values()` all'opera:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-7
//-----------------------------------------

$veg = array("pomodoro","piselli","peperone","funghi","peperoncino",
             "pomodoro","piselli","peperone","funghi",
             "pomodoro","peperone","funghi","peperoncino",
             "funghi","peperoncino","pomodoro","pomodoro","pomodoro"
             "pomodoro","piselli","peperone","funghi",
             "pomodoro","peperone","funghi","pomodoro","piselli",
             "pomodoro","piselli","pomodoro","piselli");
$count = array_count_values($veg);
echo("<tr>");
foreach($count as $vegItem=>$vegCount)
     echo("<td align='center'><img
src='graphics/$vegItem.jpg'><br>$vegCount</td>");
echo("</tr>");

?>
</table>
```

Lo script dichiara un array `$veg` costituito da ortaggi. Viene chiamata la funzione `array_count_values()` e questa crea un array di nome `$count`, che contiene il numero corrispondente di ogni ortaggio memorizzato nell'array `$veg`. Quindi il contenuto dell'array `$count` viene visualizzato con un ciclo `foreach`. Osservate che l'array `$count` utilizza come indice i nomi degli ortaggi presi dall'array `$veg`. L'output generato dallo script precedente è quello della Figura 16.7.

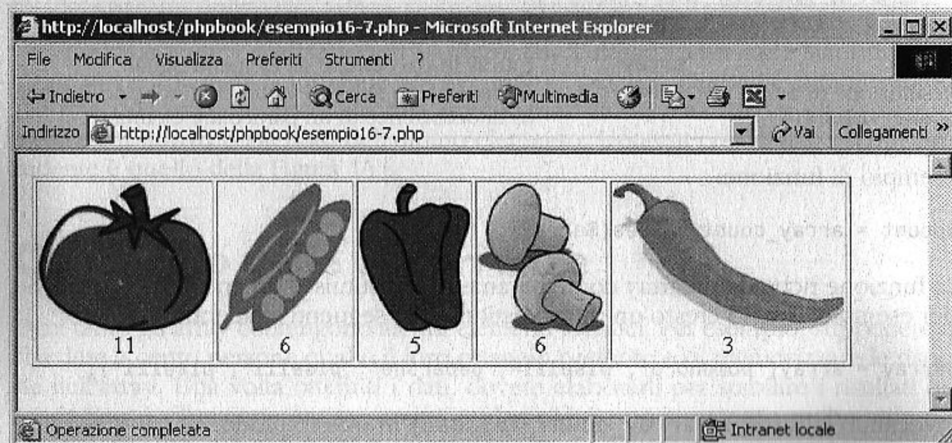


Figura 16.7
Conteggio delle occorrenze in un array.

Ordinamento

Le funzioni `sort()` e `rsort()` servono per ordinare un array. La funzione `sort()` viene utilizzata per ordinare un array in base a una sequenza numerica o alfabetica, dal valore più basso al più alto. La funzione `rsort()` ordinerà l'array dal valore più alto al più basso. La sintassi delle due funzioni è indicata di seguito.

```
void sort(array Array)
void rsort(array Array)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
Array	array	L'array da ordinare.
Restituzione di `sort()`	void	Non restituisce alcunché.
Restituzione di `rsort()`	void	Non restituisce alcunché.

Esempi di funzioni:

```
sort($letters);
rsort($letters);
```

Entrambe le funzioni richiedono un array come parametro. Lo script che segue mostra l'utilizzo delle due funzioni:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-8
//-----------------------------------------

$letters = array("b","d","c","a","i","f","e","h","g");

echo("<tr>");
foreach($letters as $letter)
    echo("<td align='center'><img src='graphics/$letter.jpg'></td>");
echo("</tr>");
sort($letters);
echo("<tr>");
foreach($letters as $letter)
    echo("<td align='center'><img src='graphics/$letter.jpg'></td>");
echo("</tr>");
rsort($letters);
echo("<tr>");
foreach($letters as $letter)
    echo("<td align='center'><img src='graphics/$letter.jpg'></td>");
echo("</tr>");

?>
</table>
```

Lo script dichiara un array di nome `$letter`, costituito da lettere ordinate in sequenza casuale. Il contenuto dell'array viene visualizzato, ordinato e visualizzato di

nuovo. Quindi, l'array viene ordinato in senso inverso e infine visualizzato. Per rendere più interessante l'output dello script è stata creata una serie di immagini che rappresentano le lettere presenti nell'array. La Tabella 16.1 illustra queste immagini e i nomi di file corrispondenti.

Tabella 16.1 Immagini delle lettere

A	B	C	D	E	F	G	H	I
a.jpg	b.jpg	c.jpg	d.jpg	e.jpg	f.jpg	g.jpg	h.jpg	i.jpg

La Figura 16.8 mostra l'output creato dallo script precedente.

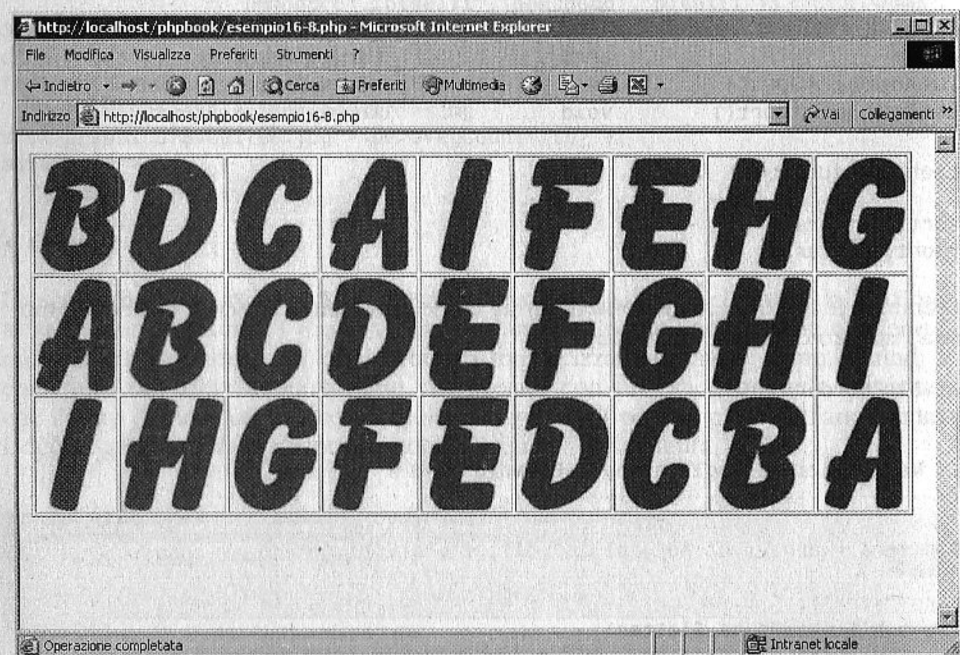


Figura 16.8
Array ordinati.

Ordinamento multiplo

La funzione `array_multisort()` può essere utilizzata per ordinare più array contemporaneamente, o un solo array multidimensionale.
La sintassi della funzione è la seguente.

```
bool array_ multisort(array Array, mixed flag)
```


La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla fun
zione.

Nome	Tipo	Descrizione
Array	array	L'array da ordinare.
flag	array	SORT_ASC ordina in sequenza crescente. SORT_DESC ordina in sequenza decrescente SORT_REGULAR confronta gli elementi nor malmente. SORT_NUMERIC confronta gli elementi i modo numerico. SORT_STRING confronta gli elementi com stringhe.
Restituzione di array_multisort()	bool	Restituisce TRUE se l'operazione di ordina mento è riuscita o FALSE se si è verificat un errore.

Esempio di funzione:

```
array_multisort($veg[0],SORT_ASC, SORT_STRING,
$veg[1],SORT_NUMERIC);
```

La funzione `array_multisort()` restituisce TRUE o FALSE in base all'esito positivo o negativo dell'operazione di ordinamento. Esaminiamo un esempio dell'utilizzo d tale funzione:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-9
//-------------------------------------

$veg = array( array("pomodoro","piselli","peperone","funghi",
             "peperoncino","pomodoro","piselli","peperone","funghi",
             "funghi","peperoncino","pomodoro","pomodoro","pomodoro",
             "pomodoro","piselli","pomodoro","piselli"),
              array(10,20,30,10,20,25,35,40,35,45,50,20,25,30,
              25,30,20,30));

echo("<tr>");
foreach($veg[0] as $vegItem)
   echo("<td align='center'><img src='graphics/$vegItem.jpg'
width='25' height='25'></td>");
echo("</tr>");
echo("<tr>");
foreach($veg[1] as $vegPrice)
   echo("<td align='center'>$vegPrice</td>");
echo("</tr>");

array_multisort($veg[0],SORT_ASC, SORT_STRING, $veg[1],SORT_NUMERIC);

echo("<tr>");
foreach($veg[0] as $vegItem)
   echo("<td align='center'><img src='graphics/$vegItem.jpg' width='25'
height='25'></td>");
echo("</tr>");
```

```
echo("<tr>");
foreach($veg[1] as $vegPrice)
   echo("<td align='center'>$vegPrice</td>");
echo("</tr>");

?>
</table>
```

Lo script inizia dichiarando un array multidimensionale di nome `$veg`, composto da due array. Tali array memorizzano il nome e il prezzo delle verdure. Vengono utilizzati due cicli `foreach` per visualizzare il contenuto dell'array in una tabella. Successivamente, viene invocata una funzione `array_multisort()`:

```
array_multisort($veg[0],SORT_ASC, SORT_STRING, $veg[1],SORT_NUMERIC);
```

Alla funzione viene passato il primo array, `$veg[0]`, che dovrà essere ordinato in sequenza crescente e in base al valore di stringa. Il secondo array, `$veg[1]`, dovrà essere ordinato in sequenza numerica. Quindi, il risultato dell'operazione di ordinamento sarà visualizzato con due cicli `foreach`. L'output dello script precedente è visibile nella Figura 16.9. È importante sottolineare che la funzione `array_multisort()` non ordina il primo array e poi il secondo separatamente, dando come risultato ortaggi con prezzi inesatti. Infatti, le relazioni intercorrenti tra i dati di ciascun array vengono mantenute. L'operazione di ordinamento del secondo array consente di visualizzare tutti gli ortaggi di un certo tipo in base al prezzo.



Figura 16.9
Output dell'operazione di ordinamento multiplo.

explode e implode

Le ultime due funzioni introdotte nel capitolo sono: `explode()` e `implode()`, che permettono di convertire un array in una stringa delimitata e riconvertire la stringa in un array. Un utente potrebbe voler memorizzare l'array in un database o in un file, e questo è un metodo pratico per estrarre e memorizzare i dati sotto forma di stringa. Inoltre, più avanti, si vedrà che il passaggio degli array all'interno dei moduli comporta problemi e una delle possibili soluzioni è passare l'array sotto forma di stringa.
La sintassi delle funzioni è indicata di seguito.

```
array explode(string Separatore, string Stringa)
string implode(string Separatore, array Array)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
Array	*array*	L'array da implodere.
Stringa	string	La stringa da esplodere.
Separatore	string	Il carattere da utilizzare per separare i contenuti dell'array.
Restituzione di `explode()`	array	Un array di elementi esplosi.
Restituzione di `implode()`	string	Una string di elementi dell'array.

Esempi di funzioni:

```
$string = implode("¦",$veg);
$newVeg = explode("¦",$string);
```

Lo script che segue illustra l'implosione di un array in una stringa che viene poi visualizzata.
La stringa viene poi riesplosa in un nuovo array e anch'esso viene visualizzato:

```
<table border='1'>
<?php

// Manipolazione degli array - Esempio 16-10
//-------------------------------------

$veg = array(0=>"pomodoro", 1=>"piselli", 2=>"peperone", 3=>"funghi",
      4=>"peperoncino");

$string = implode("¦",$veg);
echo("<tr><td align='center' colspan='5'>$string</td></tr><tr>");
$newVeg = explode("¦",$string);

foreach($newVeg as $vegItem)
   echo("<td><img src='graphics/$vegItem.jpg'></td>");
echo("</tr>");

?>
</table>
```

L'output dello script precedente è mostrato nella Figura 16.10.

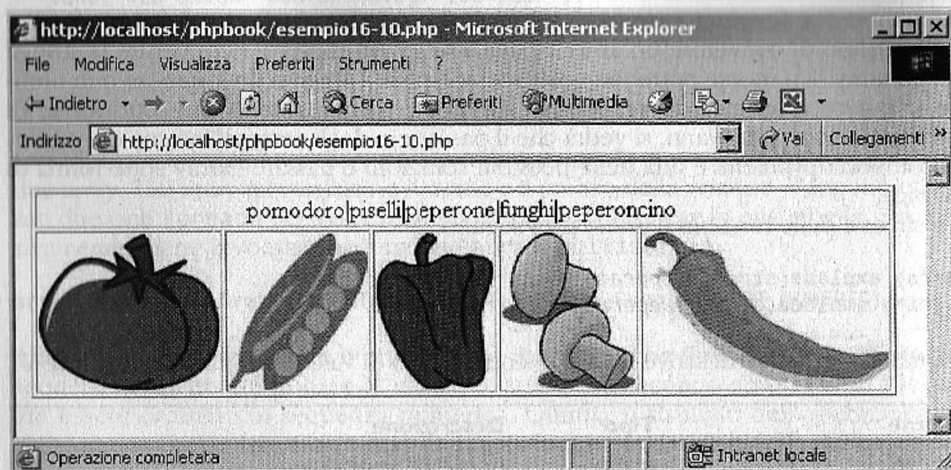


Figura 16.10
Esplosione e implosione degli array.

Riepilogo

Questo capitolo ha ripreso il concetto di array, analizzando numerose funzioni con le quali è possibile manipolare in modi diversi i dati contenuti in queste strutture. Nel prossimo capitolo torneremo al tipo di dati stringa, introdotto nel Capitolo 8, e vedremo quali funzioni è possibile utilizzare per manipolare le stringhe.

Capitolo 17

Manipolazione delle stringhe

Introduzione

Nel Capitolo 8 abbiamo introdotto il concetto di stringa e abbiamo mostrato come dichiararle e utilizzarle. Le stringhe sono molto utilizzate in PHP ed esistono diverse funzioni create appositamente per aiutare l'utente a manipolarle. In questo capitolo presenteremo alcune di queste funzioni e ne spiegheremo l'utilizzo.

Calcolo della lunghezza di una stringa

Probabilmente, la funzione per stringhe più comune è `strlen()`, che consente di contare il numero di caratteri di una stringa.
La sintassi della funzione è la seguente.

```
int strlen(string inputString);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
inputString	string	La stringa della quale si conta il numero di caratteri.
Restituzione di `strlen()`	int	Il numero di caratteri contenuti nella stringa.

Esempio di funzione:

```
$size = strlen($title);
```

L'esempio che segue utilizza `strlen()` per controllare il titolo di un messaggio inviato a un forum. I titoli dei messaggi non devono superare i 60 caratteri, pertanto occorre impedire agli utenti di inviare titoli che superino la lunghezza consentita.

```
<?php

// Manipolazione delle stringhe - Esempio 17-1
//.................................
```

```
$title = "Questo è il mio messaggio";
$size = strlen($title);

if($size <= 60) {
        echo "La lunghezza del titolo è $size. Accettato";
}
else {
        echo "La lunghezza del titolo è Ssize. Non accettato";
}

?>
```

Per prima cosa, questo script esegue una verifica della variabile `title`, mediante la funzione `strlen()`:

```
$title = "Questo è il mio messaggio";
$size = strlen($title);
```

Quindi, verifica la dimensione della stringa e visualizza un messaggio appropriato:

```
if($size <= 60) {
        echo "La lunghezza del titolo è $size. Accettato";
}
else {
        echo "La lunghezza del titolo è $size. Non accettato";
}
```

L'output dello script precedente è quello della Figura 17.1.

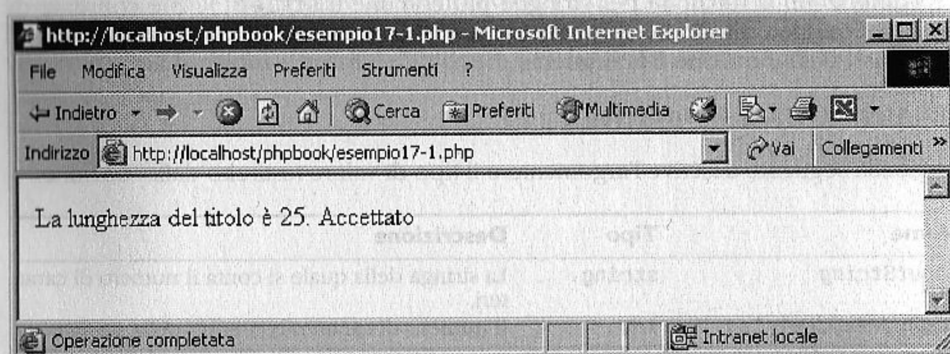


Figura 17.1
Verifica della lunghezza di una stringa.

Esplosione/implosione di parti ed elementi della stringa

La funzione `explode()` serve per creare un array di elementi partendo dalle parti di una stringa divise da un carattere separatore. Il separatore può essere un singolo carattere/numero oppure una stringa. La funzione `implode()` è l'esatto contrario di

`explode()` e viene utilizzata per creare una stringa partendo dagli elementi di un array divisi da un separatore. Nel Capitolo 16 si trovano tutti i dettagli relativi a queste due funzioni, la cui sintassi è riportata di seguito.

```
Array explode(string Separatore, string Stringa);
String implode(string Separatore, array Array);
```

Nell'esempio seguente è riportata una stringa composta da nomi propri: si vuole creare una piccola tabella in cui ciascun nome occupi una riga diversa:

```
<?php

// Manipolazione delle stringhe - Esempio 17-2
//-----------------------------------------------

$names = "mike simon david mary";
$list  = explode(" ",$names);
$rows  = count($list);
echo "<table border=1 width=220><tr>";
echo "<td bgcolor='#999999'> Nomi</td></tr>";
for($i=0;$i<$rows;$i++){
  echo "</tr><td> $list[$i] </td></tr>";
}
echo "</table>";

?>
```

Innanzitutto lo script definisce una stringa e la esplode in un array:

```
$names = "mike simon david mary";
$list = explode(" ",$names);
```

quindi, conta il numero di righe nell'array:

```
$rows = count($list);
```

Infine, utilizza un ciclo `for` per costruire la tabella che visualizzerà gli elementi di ogni array su righe diverse:

```
echo "<table border=1 width=220><tr>";
echo "<td bgcolor='#999999'> Nomi</td></tr>";
for($i=0;$i<$rows;$i++){
  echo "</tr><td> $list[$i] </td></tr>";
}
echo "</table>";

?>
```

L'output dello script è quello della Figura 17.2.

Ricerca di una stringa all'interno di un'altra

La funzione `strstr()` può essere utilizzata per trovare una parte specifica di una stringa (persino un singolo carattere) all'interno di un'altra stringa. Utilizzando un

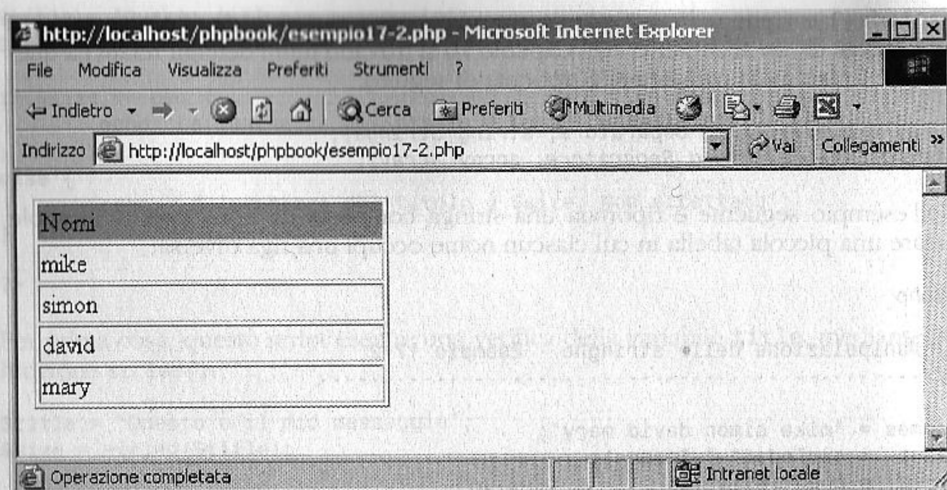


Figura 17.2
Esempio di esplosione.

identificatore `Identificatore`, la funzione restituisce la parte della stringa `inputString` dalla prima occorrenza di `Identificatore` alla fine della stringa di input. La sintassi della funzione è la seguente.

```
string strstr (string inputString, string Identificatore)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
inputString	string	La stringa da utilizzare.
Identificatore	string	Identificatore di stringa.
Restituzione di `strstr()`	string	Parte di una stringa.

Esempio di funzione:

```
$email = 'user@example.com';
$domain = strstr($email, '@');
```

Lo script PHP che segue illustra l'utilizzo della funzione:

```
<?php

// Manipolazione delle stringhe - Esempio 17-3
//-----------------------------------------------------

$email = 'user@example.com';
$domain = strstr($email, '@');
echo "$email<br>$domain";

?>
```


L'output dello script è il seguente:

```
user@example.com
@example.com
```

Sostituzione di una parte di una stringa

In genere, la funzione `str_replace()` viene utilizzata per cambiare le parti di una stringa in un'altra.
La sintassi della funzione è la seguente.

```
string str_replace (string parteCercata, string parteSostituita, string Inizio)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
parteCercata	string	Stringa da cercare.
parteSostituita	string	Stringa con cui sostituire *parteCercata*.
Inizio	string	Stringa su cui eseguire le operazioni.
Restituzione di `str_replace()`	string	Stringa modificata.

Esempio di funzione:

```
$text = "Ciao, questo è [b]testo in grassetto[/b]";
$text = str_replace("[b]","<b>",$text);
```

Considerate lo script seguente che utilizza questa funzione:

```
<?php

// Manipolazione delle stringhe - Esempio 17-4
//-----------------------------------------------------

$text = "Ciao, questo è [b]testo in grassetto[/b]";
echo "$text";

$text = str_replace("[b]","<b>",$text);
$text = str_replace("[/b]","</b>",$text);

echo "<br><br>";
echo "$text";

?>
```

Nello script precedente si devono convertire i simboli `[b]` e `[/b]` della variabile `$text` in `<b>` e `</b>`, in modo che il testo possa comparire in grassetto all'interno del browser Web. Innanzitutto, lo script cerca la parte `[b]` della stringa e la sosti-

tuisce con un tag . Poi, la stringa [/b] viene sostituita con . La Figura 17.3 mostra l'output prodotto da questo script.

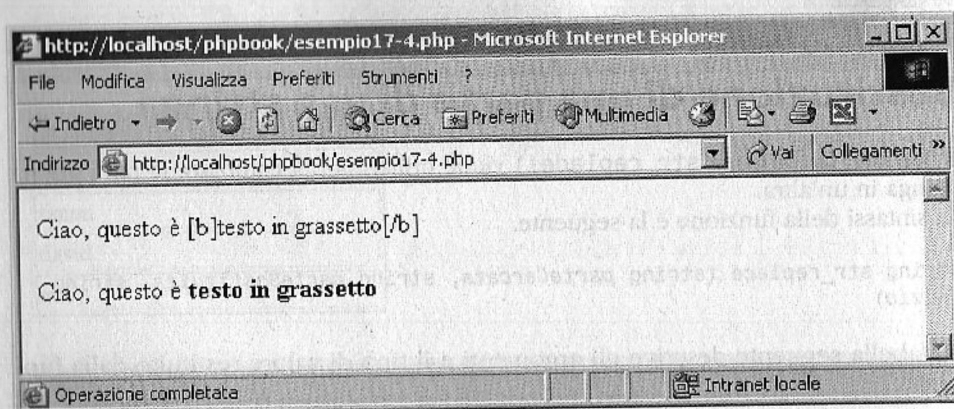


Figura 17.3
Sostituzione di una parte della stringa.

Ripetizione e inversione di una stringa

Vi piacerebbe avere la possibilità di ripetere una stringa in modo automatico senza continui interventi manuali, oppure di invertire una stringa? Per queste situazioni sono disponibili due funzioni, la cui sintassi è riportata di seguito.

```
string str_repeat ( string inputString, int Moltiplicatore);
string strrev ( string inputString);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
inputString	string	La stringa da utilizzare.
Moltiplicatore	int	Il numero di volte per cui si deve ripetere la stringa.
Restituzione di str_repeat()	string	Restituisce la stringa di input, ripetuta tante volte quante ne sono definite dal moltiplicatore.
Restituzione di strrev()	string	Restituisce la stringa di input invertita.

Esempi di funzioni:

```
$repeated = str_repeat ( $inputString, 5 );
$reverse = strrev ( $inputString );
```


L'esempio seguente crea una tabella con 10 colonne. Ciò è possibile ripetendo la stringa <td>spazio</td> dieci volte, come mostrato di seguito:

```
<?php

// Manipolazione delle stringhe - Esempio 17-5
//---------------------------------------------------

echo "<table border='1' height=10>";
echo "<tr>";

echo str_repeat("<td width=50> </td>",10);

echo "</tr>";
echo "</table>";

?>
```

L'output dell'esempio è quello della Figura 17.4.

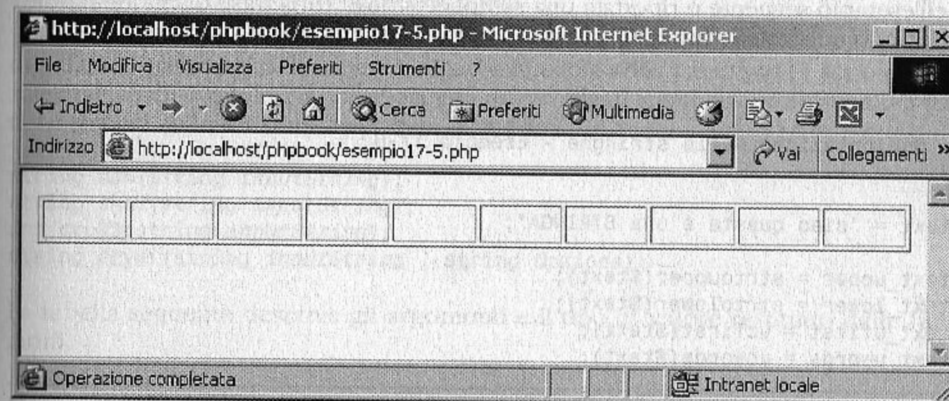


Figura 17.4
Ripetizione di parte di una stringa (<td>spazio</td>).

Conversione maiuscolo/minuscolo di una stringa

Succede talvolta di dover convertire da minuscolo a maiuscolo, o viceversa, qualche stringa utilizzata negli script. Le funzioni riportate di seguito sono molto utili, poiché aiutano a modificare i caratteri minuscoli o maiuscoli delle parole contenute nelle stringhe o ad alterare il primo carattere di ogni parola della stringa.
La sintassi delle funzioni è riportata di seguito.

```
string strtoupper(inputString);
string strtolower(inputString);
string ucfirst(inputString);
string ucwords(inputString);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
inputString	string	La stringa da utilizzare
Output di strtoupper ()	string	Converte la stringa di input in caratteri maiuscoli.
Output di strtolower ()	string	Converte la stringa di input in caratteri minuscoli.
Output di ucfirst ()	string	Volge al maiuscolo il primo carattere della stringa di input.
Output di ucwords ()	string	Volge al maiuscolo il primo carattere di ogni parola contenuta nella stringa di input

Esempi di funzioni:

```
$text_upper = strtoupper($text);
$text_lower = strtolower($text);
$text_ufirst = ucfirst($text);
$text_uwords = ucwords($text);
```

Nell'esempio seguente è riportata una semplice stringa (una frase di cinque parole) e saranno utilizzate le funzioni per trasformare la stringa in diversi modi:

```
<?php

// Manipolazione delle stringhe - Esempio 17-6
//-----------------------------------------------

$text = "ciao questa è una STRINGA";

$text_upper = strtoupper($text);
$text_lower = strtolower($text);
$text_ufirst = ucfirst($text);
$text_uwords = ucwords($text);

echo "Testo originale: $text <br><br>";
echo "Maiuscolo: $text_upper <br>";
echo "Minuscolo: $text_lower <br>";
echo "Primo carattere maiuscolo: $text_ufirst <br>";
echo "Primi caratteri tutti maiuscoli: $text_uwords";

?>
```

L'output dello script precedente è quello della Figura 17.5.

Cifratura delle stringhe

Esistono più di dieci metodi per cifrare le stringhe. La cifratura può essere a una via (non esiste la possibilità di decifrare la stringa) e a due vie (la parola cifrata può essere decifrata con il metodo del dizionario o altri metodi). In questo paragrafo saranno presentati i quattro metodi di cifratura più utilizzati. Alcuni di essi vengono impiegati spesso per verificare l'integrità di file e dati. MD5() viene utilizzata soprattutto per cifrare le password nelle applicazioni Web, mentre Crypt() serve per crittografare le password del sistema Linux. Le funzioni md5() e crypt() sono analizzate anche nel Capitolo 20, che si occupa dei problemi legati alla protezione dei dati. La sintassi delle funzioni è riportata di seguito.

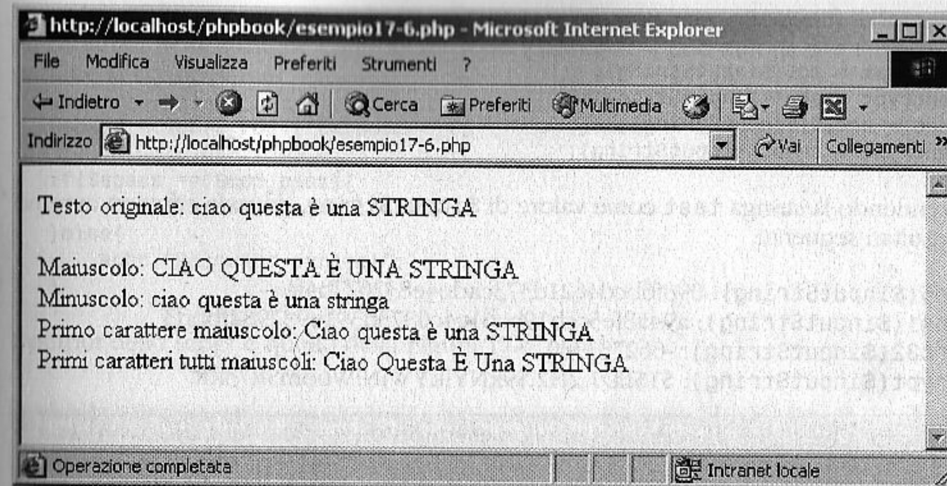


Figura 17.5
Modifica delle lettere maiuscole/minuscole della stringa.

```
string md5(string inputString);
string sha1(string inputString);
int crc32(string inputString);
string crypt(string inputString , string Opzione);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
inputString	string	La stringa da utilizzare.
Opzione	string	CRYPT_STD_DES. Cifratura DES standard in blocchi di due caratteri. CRYPT_EXT_DES. Cifratura DES estesa in blocchi di nove caratteri. CRYPT_MD5. Cifratura MD5 in blocchi di 12 caratteri che inizia con 1. CRYPT_BLOWFISH. Cifratura Blowfish in blocchi di 16 caratteri che inizia con 2.
Restituzione di md5()	string	Una stringa lunga 32 bit contenente la stringa di input cifrata.
Restituzione di sha1()	string	Una stringa di 40 caratteri contenente la stringa di input cifrata.
Restituzione di crc32()	int	Una stringa contenente il polinomio crc32 della stringa di input.
Restituzione di crypt()	string	Una stringa di 40 caratteri contenente la stringa di input cifrata.

Esempi di funzioni:

```
$encrypt = md5($inputString);
$encrypt = sha1($inputString);
$crc = crc32($inputString);
$encrypt = crypt($inputString);
```

Prendendo la stringa `test` come valore di `$inputString`, dopo la cifratura avremo i risultati seguenti:

`md5($inputString)`: 098f6bcd4621d373cade4e832627b4f6
`sha1($inputString)`: a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
`Crc32($inputString)`: -662733300
`Crypt($inputString)`: 1LD1.2H2.$9kNYlkYWiNiWbomSR73kR.

Lo script che segue illustra un semplice esempio di verifica della password con il metodo della codifica MD5:

```
<?php

// Manipolazione delle stringhe - Esempio 17-7
//-------------------------------------------------------

$my_pass  = "377729";
$md5_pass = md5($my_pass);

if(isset($_POST["submit"])){
   $spass = md5($_POST["password"]);

   if($spass == $md5_pass){
      echo "Password accettata!";
   }else{
      echo "Password errata!";
   }
}

?>

<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <form name="form1" method="post"
       action="<?php echo $_SERVER["PHP_SELF"]; ?>">

      Verifica password:
      <input type="text" name="password">
      <input type="submit" name="submit" value="Invia">

      </form>
   </body>
</html>
```

In questo script abbiamo cifrato la password **377729** con il metodo MD5:

```
$my_pass = "377729";
$md5_pass = md5($my_pass);
```


Lo script cifra la password inoltrata (**377728**) e la confronta con quella cifrata in precedenza. Se le stringhe confrontate sono identiche, si può autorizzare l'utente a stabilire la connessione:

```
$spass = md5($_POST["password"]);

if($spass == $md5_pass){
   echo "Password accettata!";
}else{
   echo "Password errata!";
}
```

L'output dello script è quello della Figura 17.6.

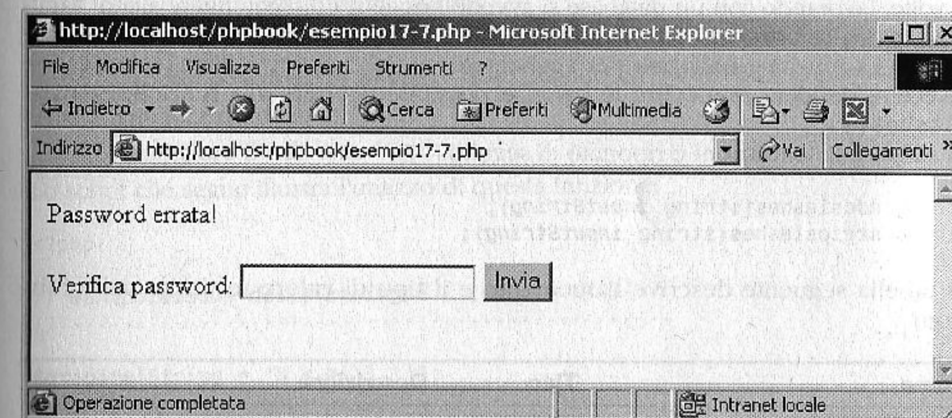


Figura 17.6
Semplice esempio di verifica della password.

Conteggio delle parole

La funzione `str_word_count()` conta il numero di parole contenute in una stringa. La sintassi della funzione è la seguente.

```
mixed str_word_count(string inputString , int Modalità)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
inputString	`string`	La stringa da utilizzare.
Modalità	`int`	1 restituisce un array contenente tutte le parole trovate all'interno della stringa; 2 restituisce un array associativo, in cui la chiave è la posizione numerica della parola nella stringa e il valore è la parola stessa.
Restituzione di `str_word_count()`	`mixed`	Se la modalità non è specificata, visualizza il numero totale di parole presenti nella stringa di input.

Esempio di funzione:

```
$count = str_word_count($string,1);
```

Se non si specifica alcuna modalità, il risultato sarà un numero che indica quante parole sono state trovate all'interno della stringa; se la modalità è impostata a "1", il risultato sarà un array contenente tutte le parole trovate all'interno della stringa, mentre se la modalità è impostata a "2", il risultato sarà un array associativo, in cui la chiave è la posizione numerica della parola nella stringa e il valore è la parola stessa.

Aggiunta e rimozione delle barre

A volte, lavorando con un database si avverte l'esigenza di aggiungere alcuni caratteri, come l'apostrofo ('), le virgolette ("), le barre contrarie (\) e NULL (il byte NULL). La funzione utilizzata per aggiungere questi caratteri è `addslashes()`. Se invece si desidera rimuovere le barre, è possibile ricorrere alla funzione `stripslashes()`.
La sintassi delle funzioni è riportato di seguito.

```
string addslashes(string inputString);
string stripslashes(string inputString);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
inputString	string	Stringa su cui eseguire le operazioni con le barre.
Restituzione di `addslashes()`	string	Stringa con i caratteri di escape.
Restituzione di `stripslashes()`	string	Stringa con le barre rimosse.

Esempi di funzioni:

```
$escapedString = addslashes($inputString);
$normalString = stripslashes($inputString);
```

Conversione di caratteri speciali in entità HTML

In HTML alcuni caratteri hanno un significato particolare e dovrebbero essere rappresentati da entità HTML, se si desidera conservarlo. La funzione `htmlspecialchars()` restituisce una stringa con questi caratteri convertiti nei rispettivi codici HTML.
La sintassi della funzione è la seguente.

```
string htmlspecialchars(string inputString);
```


La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
inputString	string	Stringa su cui eseguire le operazioni.
Restituzione di `htmlspecialcharacters()`	string	Stringa con codifica HTML.

Esempio di funzione:

```
$htmlEncoded = htmlspecialchars($inputString);
```

Le conversioni eseguite sono le seguenti:

- `&` (e commerciale) diventa `'&'`.
- `"` (virgolette) diventa `'"'` quando `ENT_NOQUOTES` non è impostato.
- `'` (apostrofo) diventa `'''` solo quando `ENT_QUOTES` è impostato.
- `<` (minore di) diventa `'<'`.
- `>` (maggiore di) diventa `'>'`.

Lo script che segue illustra l'utilizzo di questa funzione:

```
<?php

// Manipolazione delle stringhe - Esempio 17-8
//-------------------------------------------------

$originalString = "& < > \" \'";

$htmlEncoded = htmlspecialchars($originalString);

echo "Originale: $originalString <br> Codificata: $htmlEncoded";

$len = strlen($originalString);
$len2 = strlen($htmlEncoded);

echo "<br>Lunghezza originale: $len <br> Lunghezza codificata: $len2";

?>
```

L'output di questo script è quello della Figura 17.7.
Notate che nell'esempio della Figura 17.7 l'output di testo è lo stesso in entrambi i casi. Ciò dipende dal fatto che un browser Web visualizza una "&" e una "&amp" come una "&". Tuttavia si può affermare che la conversione è avvenuta perché la lunghezza delle due stringhe è diversa.

Confronto di stringhe con diverse impostazioni di maiuscolo/minuscolo

A volte è necessario mettere a confronto stringhe diverse per vedere se contengono le stesse parole, a prescindere dalle differenze di maiuscole/minuscole. Per esem-

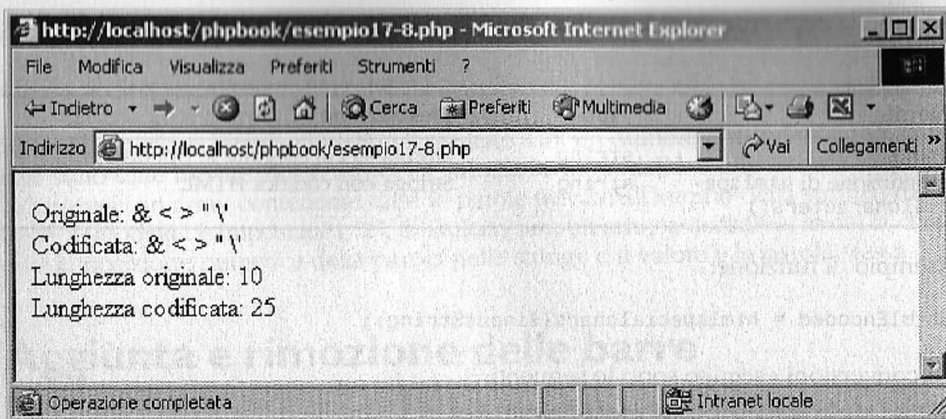


Figura 17.7
Esempi di caratteri speciali HTML.

pio, si vuole poter dire che la stringa "Simon" è identica a "sIMON". A tal fine è possibile utilizzare la funzione `strncasecmp()`.
La sintassi della funzione è la seguente.

```
string strncasecmp(string Stringa1, string Stringa2, int Lunghezza);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Stringa1	`string`	Prima stringa da confrontare.
Stringa2	`string`	Seconda stringa da confrontare.
Lunghezza	`int`	Numero dei caratteri in ciascuna stringa da utilizzare nel confronto.
Restituzione di `strncasecamp()`	`int`	Restituisce 0 se le stringhe sono uguali.

Esempio di funzione:

```
$equal = strncasecmp("Simon", "sIMON", 5);
```

Riepilogo

Questo capitolo ha ripreso il concetto di stringa. Sono state analizzate numerose funzioni con le quali è possibile manipolare in modi diversi i dati contenuti nelle stringhe. Le funzioni qui presentate sono quelle più utilizzate dai programmatori, tuttavia esistono più di venti funzioni di stringa diverse utilizzabili per applicazioni e script più specifici. Nel prossimo capitolo verrà introdotto il concetto dell'interazione con l'utente; si spiegherà in che modo le applicazioni PHP possano interagire con gli utenti finali e le operazioni disponibili.

Parte IV

Interazione con l'utente

Capitolo 18

Interazione con l'utente

Introduzione

I programmi che non offrono alcuna forma di interazione con gli utenti hanno possibilità di impiego molto limitate. PHP è in grado di interagire con un utente mediante una pagina Web che utilizza i moduli. I moduli consentono di inserire i dati e inoltrarli al server cosicché possano essere elaborati da PHP. Esistono moduli di ogni forma e dimensione; alcuni sono facili da identificare in quanto tali, altri sono meno ovvi. In questo capitolo introdurremo i principi fondamentali e mostreremo come combinarli per consentire interazioni piuttosto complesse.

PHP e i moduli

I moduli sono un componente di HTML, non di PHP. Anche se PHP può visualizzare i moduli nello stesso modo in cui visualizza tutti gli altri elementi HTML, in realtà non aggiunge nulla di nuovo alle funzionalità offerte dal linguaggio HTML standard. PHP può essere però utilizzato per controllare i moduli ed elaborare i dati forniti dall'utente nell'interazione con il modulo, un'operazione che HTML non è in grado di fare. Dato che PHP è un linguaggio di scripting sul lato server, i dati dei moduli devono essere trasmessi al server per essere elaborati, dopodiché l'output generato dovrà essere nuovamente inviato all'utente. Una differenza notevole rispetto alle modalità di utilizzo di JavaScript, dove l'elaborazione può essere eseguita sul computer del client all'interno del browser.

Un modulo semplice

Il primo passo sarà la creazione di un modulo HTML standard e la specificazione di alcuni elementi dell'immissione di dati. Considerate lo script seguente:

```
<html>

<!- Interazione con l'utente - Esempio 18-1 ->
<!- ---------------------------------- ->
```

```
  <body>
    <h2>Inserire i propri dati personali:</h2>
    <form action='esempio18-2.php' method='post'>

    Nome: <input type='text' name='firstname'>
    <br>
    Cognome: <input type='text' name='surname'>
    <br>
    Nome utente: <input type='text' name='username'>
    <br>
    Password: <input type='password' name='password'>
    <br>
    <input type='submit'>

    </form>
  </body>
</html>
```

Questo script è un documento HTML, non uno script PHP, pertanto deve essere salvato con l'estensione .html. Lo script mostra gli elementi fondamentali di un modulo HTML. Innanzitutto, il modulo inizia con un elemento `<form>`, che ha due attributi chiave: `action` e `method`. L'attributo `action` specifica dove inviare i dati del modulo per l'elaborazione: potrebbe trattarsi un'applicazione CGI scritta con qualunque linguaggio di programmazione, oppure potrebbe essere uno script PHP. L'attributo `method` specifica come inviare i dati del modulo all'applicazione. I metodi principali sono due, `POST` e `GET`. PHP supporta `POST`, quindi è necessario utilizzare questo metodo.
Nell'esempio, l'elemento del modulo specifica che lo script per elaborare i dati del modulo si chiama esempio18-2.php, che però non esiste ancora:

```
<form action='esempio18-2.php' method='post'>
...
</form>
```

La parte restante del modulo è costituita da quattro campi per l'immissione dei dati, tre dei quali sono del tipo `text` e uno di tipo `password`:

```
    Nome: <input type='text' name='firstname'>
    <br>
    Cognome: <input type='text' name='surname'>
    <br>
    Nome utente: <input type='text' name='username'>
    <br>
    Password: <input type='password' name='password'>
    <br>
```

In pratica si tratta dello stesso tipo di campi per l'immissione dati, tranne per il fatto che i campi password nascondono i dati in essi contenuti visualizzando al loro posto caratteri *. Osservate che ogni campo di input dei dati ha un nome specifico, che è stato scelto per rappresentare i dati relativi a ciascun elemento. I nomi dei campi modulo sono importanti, poiché serviranno per accedere ai dati dei moduli nello script PHP. Il modulo finisce con un campo di input di tipo `submit`:

```
<input type='submit'>
```


Questo campo è necessario per consentire all'utente di inoltrare i dati del modulo dopo averlo completato. L'output dello script HTML precedente è illustrato nella Figura 18.1. In pratica, si tratta di ciò che HTML è in grado di fare con i moduli. Più avanti nel capitolo si vedrà che esiste qualche altro tipo di campo per i moduli HTML; se però volete utilizzare in qualche modo i dati del modulo, è necessario cominciare a creare uno script PHP che gestisca le informazioni inserite.

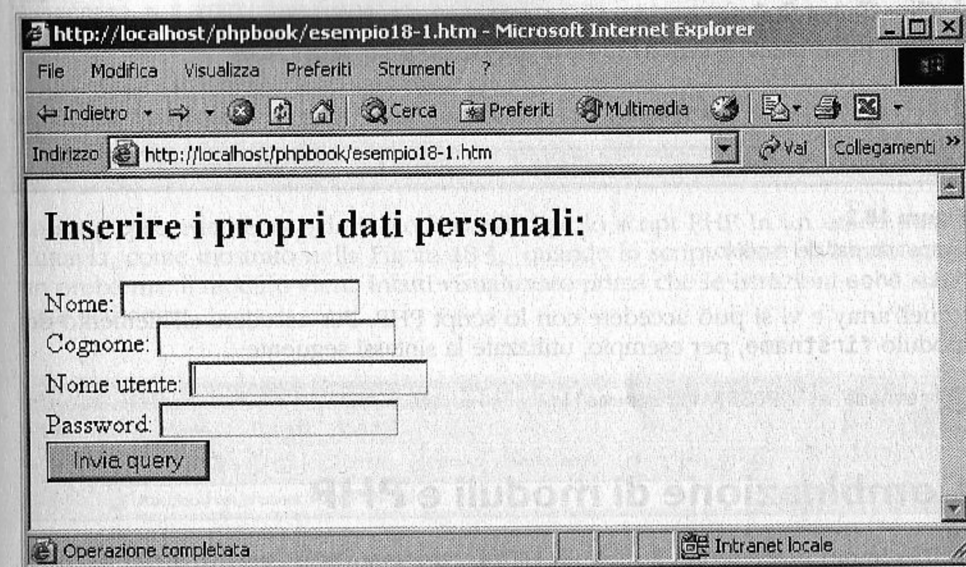


Figura 18.1
Un modulo semplice.

A questo punto create uno script PHP che gestisca i dati passati al server attraverso il modulo. Lo script è molto semplice, poiché l'unica operazione che esegue è visualizzare i dati ricevuti, come mostrato nella Figura 18.2.

```
<?php

// Interazione con l'utente - Esempio 18-2
//-----------------------------------

$firstname = $_POST["firstname"];
$surname = $_POST["surname"];
$username = $_POST["username"];
$password = $_POST["password"];

echo("Ciao $firstname $surname<br>");
echo("Il tuo nome utente è $username e la tua password è $password");
?>
```

Accedere ai dati del modulo è molto semplice. Quando si crea un modulo, contemporaneamente viene creato un array associato di variabili, che prende il nome di `$_POST`. Ogni elemento del modulo HTML con un nome univoco viene memorizza-

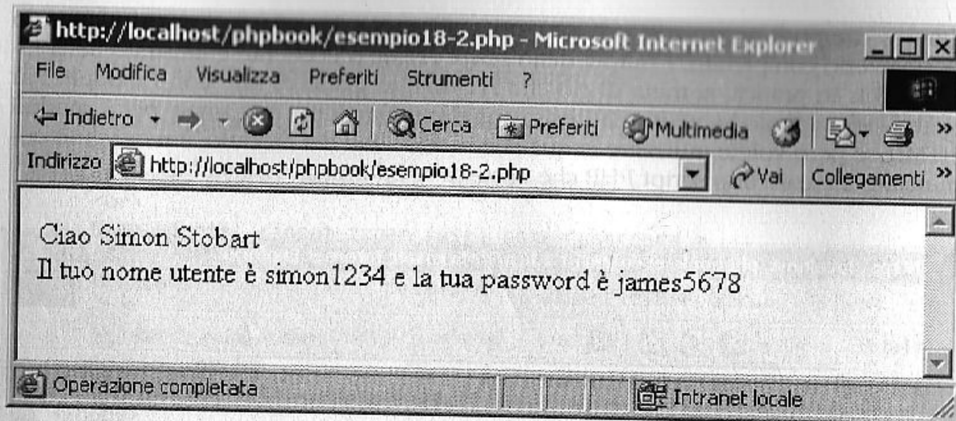


Figura 18.2
Output dei dati del modulo.

to nell'array e vi si può accedere con lo script PHP. Per accedere all'elemento del modulo `firstname`, per esempio, utilizzate la sintassi seguente:

```
$firstname = $_POST["firstname"];
```

Combinazione di moduli e PHP

Nell'esempio precedente, il modulo HTML e lo script PHP sono stati scritti come due file separati. Il primo conteneva il modulo HTML, il secondo lo script PHP necessario per elaborare i dati. Non è sicuramente il modo migliore di implementare questa soluzione, poiché è difficile fare nuovamente riferimento al modulo, soprattutto se l'utente deve reinserire qualche dato nuovo. Il metodo migliore consiste nel combinare il modulo e PHP elaborandoli insieme nello stesso script, come mostrato di seguito:

```
<html>
  <body>
    <h2>Inserire i propri dati personali:</h2>
    <form action='esempio18-3.php' method='post'>

    Nome: <input type='text' name='firstname'>
    <br>
    Cognome: <input type='text' name='surname'>
    <br>
    Nome utente: <input type='text' name='username'>
    <br>
    Password: <input type='password' name='password'>
    <br>
    <input type='submit'>

    </form>
  </body>
</html>
<?php

// Interazione con l'utente - Esempio 18-3
//-------------------------------------

$firstname = $_POST["firstname"];
$surname = $_POST["surname"];
$username = $_POST["username"];
$password = $_POST["password"];

echo("Ciao $firstname $surname<br>");
echo("Il tuo nome utente è $username e la tua password è $password");

?>
```

Lo script precedente associa il modulo HTML e lo script PHP in un unico punto. Tuttavia, come mostrato nella Figura 18.3, quando lo script viene elaborato sorge un problema: il modulo viene infatti visualizzato prima che le istruzioni `echo` siano state elaborate.

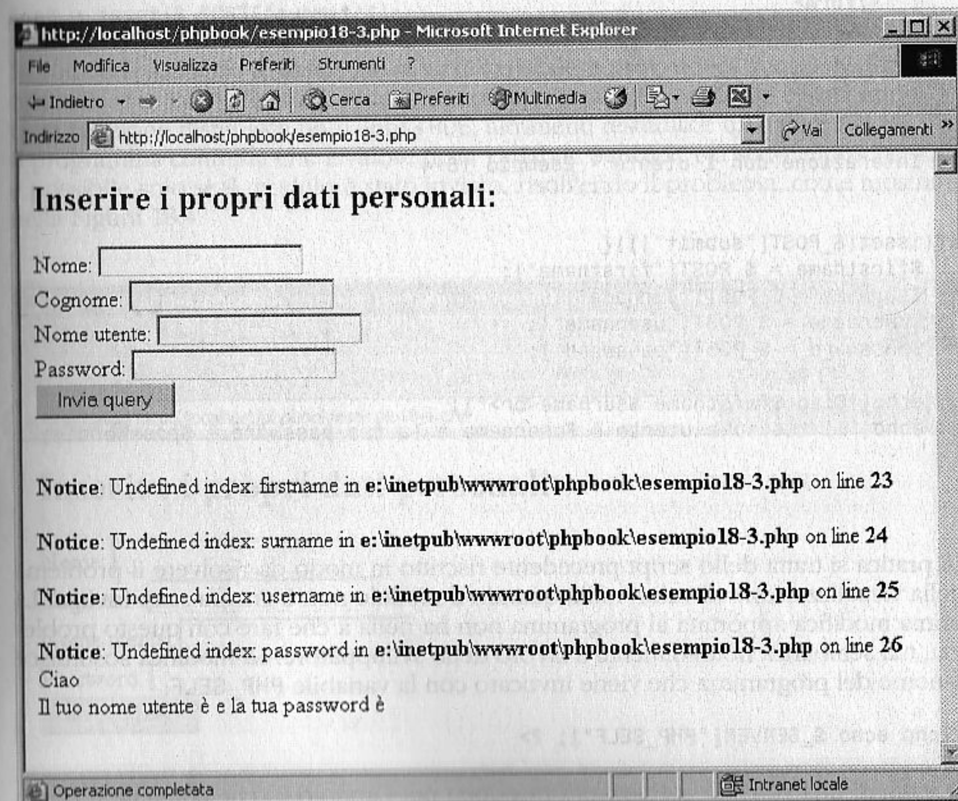


Figura 18.3
Il problema del modulo.

Dopo che il modulo è stato inviato per la prima volta, questo problema non sussiste più; tuttavia, la prima volta che si carica la pagina le variabili del modulo non contengono alcun dato, e questo comporta la visualizzazione di un messaggio incompleto. Quello di cui avete bisogno è uno strumento per stabilire se i dati del modulo siano stati o meno inviati. Considerate lo script seguente:

```
<html>
   <body>
      <h2>Inserire i propri dati personali:</h2>
      <form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>

      Nome: <input type='text' name='firstname'>
      <br>
      Cognome: <input type='text' name='surname'>
      <br>
      Nome utente: <input type='text' name='username'>
      <br>
      Password: <input type='password' name='password'>
      <br>
      <input type='submit' name='submit'>

      </form>
   </body>
</html>

<?php

// Interazione con l'utente - Esempio 18-4
//-----------------------------------

if(isset($_POST["submit"])){
   $firstname = $_POST["firstname"];
   $surname = $_POST["surname"];
   $username = $_POST["username"];
   $password = $_POST["password"];

   echo("Ciao $firstname $surname<br>");
   echo("Il tuo nome utente è $username e la tua password è $password");
}

?>
```

In pratica si tratta dello script precedente riscritto in modo da risolvere il problema della visualizzazione di valori nulli, quando il modulo non è ancora stato inviato. La prima modifica apportata al programma non ha nulla a che fare con questo problema, ma semplifica notevolmente il lavoro dello sviluppatore. La modifica sostituisce il nome del programma che viene invocato con la variabile `PHP_SELF`:

```
<?php echo $_SERVER["PHP_SELF"]; ?>
```

Si tratta di una variabile del server (introdotta nel Capitolo 9) che contiene il nome dello script corrente; questa variabile fa sì che lo script chiami sempre se stesso, a prescindere dal nome con cui è stato salvato. La modifica successiva serve per assegnare un nome al pulsante di invio:

```
<input type='submit' name='submit'>
```

In questo caso, il nome è stato impostato a **`submit`**. Infine, è stata inserita un'istruzione **`if`** per controllare il risultato della chiamata alla funzione **`isset()`**, già discusso nel Capitolo 9.
La sintassi della funzione è la seguente.

```
int isset(mixed Valore);
```

La tabella seguente descrive l'argomenti e il tipo di valore restituito dalla funzione.

Nome	**Tipo**	**Descrizione**
`Valore`	`mixed`	La variabile di cui dev'essere verificata l'esistenza.
Restituzione di `isset()`	`int`	`TRUE` se la variabile esiste, `FALSE` in caso contrario.

Esempio di funzione:

```
$set = isset($_POST["submit"]);
```

La funzione richiede il passaggio di una variabile e provvederà successivamente a controllare se tale variabile è stata o meno impostata. Se la variabile è stata impostata, la funzione restituisce un valore `TRUE`, altrimenti restituisce un valore `FALSE`.
Il programma controlla che il valore della variabile **`$submit`** sia stato impostato. Ciò è possibile solo se il modulo è stato inviato, risolvendo il problema, come mostrato nella Figura 18.4.

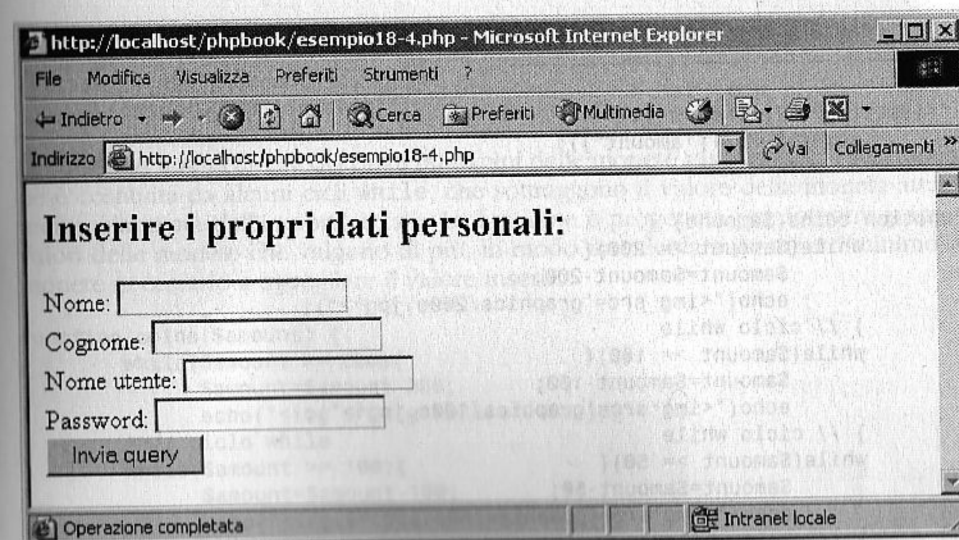


Figura 18.4
Soluzione del problema del modulo.

Calcolatore di monete

Finora gli esempi dei moduli sono stati molto semplici. Si tratta ora di creare qualcosa di più interessante. Lo script che segue illustra un modulo molto semplice associato a una funzione "utile": consente infatti a un utente di inserire un valore mediante un campo del modulo e di inviarlo. Il valore in questione rappresenta un prezzo. Per esempio, 134 rappresenta 1 euro e 34 centesimi, mentre 56 rappresenta 56 centesimi. Lo script calcola il numero minimo di monete necessario per raggiungere l'importo inserito mediante il modulo. Il programma utilizza le immagini (salvate nella directory di immagini) della Tabella 18.1.

Tabella 18.1 Immagini e nomi di file delle monete.

1c.jpg	2c.jpg	5c.jpg	10c.jpg	20c.jpg	50c.jpg	100c.jpg	200c.jpg

Osservate lo script che permette di ottenere questo risultato:

```
<h2>Calcolatore del numero minimo di monete</h2>
<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
Inserire l'importo: <input type='text' name='amount'>
<input type='submit'>
</form>

<?php

// Interazione con l'utente - Esempio 18-5
//---------------------------------------------

if(isset($_POST["amount"]))
        coins($_POST["amount"]);


function coins($amount) {
        while($amount >= 200){
              $amount=$amount-200;
              echo("<img src='graphics/200c.jpg'>");
        } // ciclo while
        while($amount >= 100){
              $amount=$amount-100;
              echo("<img src='graphics/100c.jpg'>");
        } // ciclo while
        while($amount >= 50){
              $amount=$amount-50;
              echo("<img src='graphics/50c.jpg'>");
        } // ciclo while
        while($amount >= 20){
              $amount=$amount-20;
              echo("<img src='graphics/20c.jpg'>");
        } // ciclo while
        while($amount >= 10){
              $amount=$amount-10;
              echo("<img src='graphics/10c.jpg'>");
        } // ciclo while
        while($amount >= 5){
              $amount=$amount-5;
              echo("<img src='graphics/5c.jpg'>");
        } // ciclo while
        while($amount >= 2){
              $amount=$amount-2;
              echo("<img src='graphics/2c.jpg'>");
        } // ciclo while
        if($amount > 0){
              echo("<img src='graphics/1c.jpg'>");
        }
}

?>
```

Lo script inizia visualizzando un semplice modulo costituito da un campo per l'immissione di testo e da un pulsante di invio:

```
<h2>Calcolatore del numero minimo di monete</h2>
<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>
Inserire l'importo: <input type='text' name='amount'>
<input type='submit'>
</form>
```

Innanzitutto lo script PHP controlla la variabile `$amount` per stabilire se è stato inserito un importo mediante il modulo. Se questa variabile è impostata, viene invocata la funzione `coins()`, alla quale viene passato il valore dell'importo inserito nel modulo:

```
<?php

if(isset($_POST["amount"]))
        coins($_POST["amount"]);
```

La funzione `coins()` calcola quali immagini delle monete visualizzare. Tale funzione è costituita da alcuni cicli `while`, che sottraggono il valore della moneta attualmente elaborata dall'importo totale. La funzione è progettata per sottrarre prima i valori delle monete che valgono di più, in modo da calcolare il numero minimo di monete necessario a eguagliare il valore inserito:

```
function coins($amount) {
        while($amount >= 200){
              $amount=$amount-200;
              echo("<img src='graphics/200c.jpg'>");
        } // ciclo while
        while($amount >= 100){
              $amount=$amount-100;
              echo("<img src='graphics/100c.jpg'>");
        } // ciclo while
        while($amount >= 50){
              $amount=$amount-50;
              echo("<img src='graphics/50c.jpg'>");
        } // ciclo while
```

```
        while($amount >= 20){
              $amount=$amount-20;
              echo("<img src='graphics/20c.jpg'>");
        } // ciclo while
        while($amount >= 10){
              $amount=$amount-10;
              echo("<img src='graphics/10c.jpg'>");
        } // ciclo while
        while($amount >= 5){
              $amount=$amount-5;
              echo("<img src='graphics/5c.jpg'>");
        } // ciclo while
        while($amount >= 2){
              $amount=$amount-2;
              echo("<img src='graphics/2c.jpg'>");
        } // ciclo while
        if($amount > 0){
              echo("<img src='graphics/1c.jpg'>");
        }
}
```

Alla prima esecuzione, l'output ottenuto dal programma precedente è uguale a quello illustrato nella Figura 18.5.

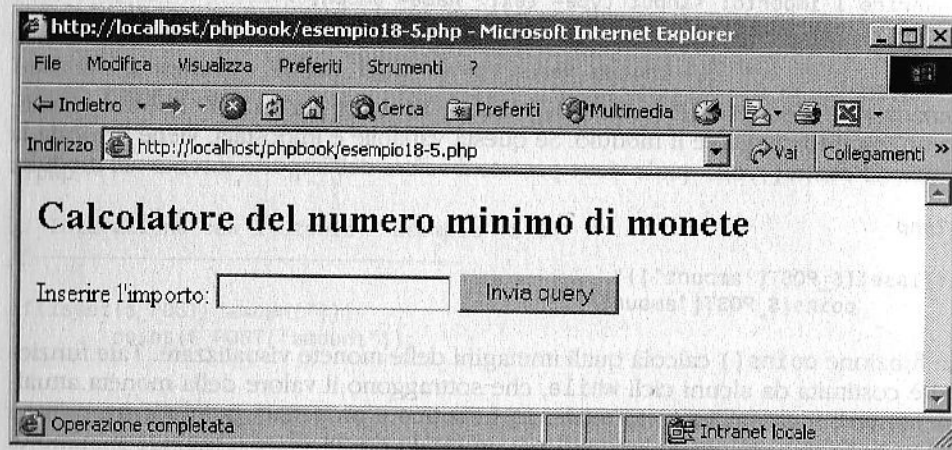


Figura 18.5
L'input dell'utente per il calcolatore di monete.

Se l'utente immette un valore nel campo modulo e fa clic sul pulsante *Invia query*, il programma calcola il numero minimo di monete necessario per raggiungere l'importo specificato. Per esempio, se l'utente digita 99 (che rappresenta 99 centesimi), ottiene l'output della Figura 18.6.

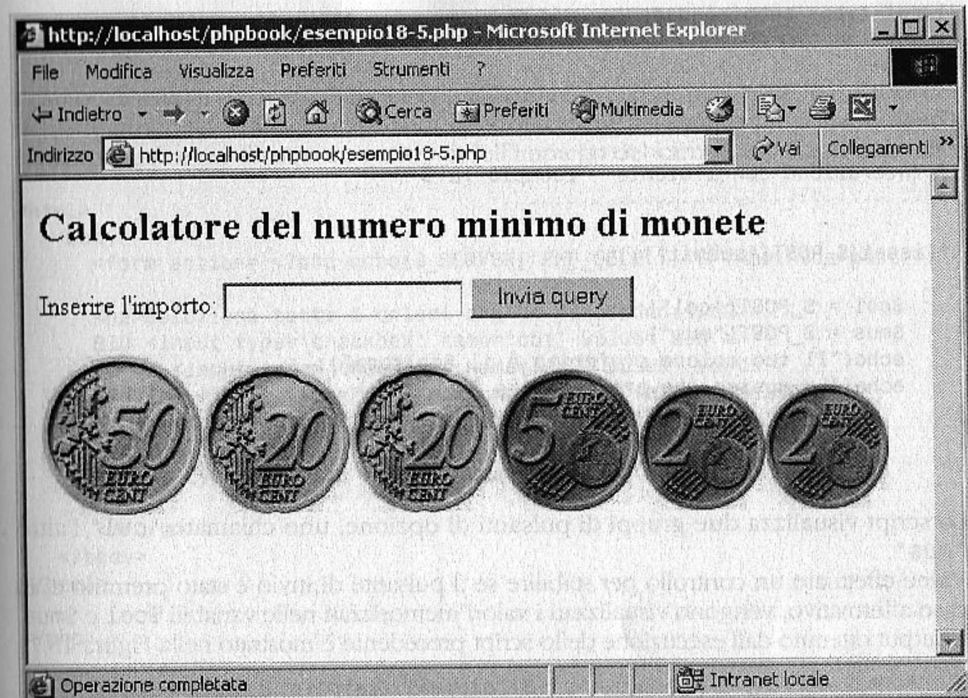


Figura 18.6
L'output del calcolatore di monete.

Pulsanti di opzione

Oltre ai campi di testo e password e al pulsante di invio introdotti finora, esistono molti altri tipi di campi. Il primo è il campo di opzione, che non consente all'utente di inserire del testo, ma fornisce una serie di pulsanti di opzione tra i quali l'utente può scegliere. Si può selezionare solo uno dei pulsanti raggruppati. Per essere certi che il modulo sappia quali pulsanti formano un gruppo, tutti i pulsanti raggruppati devono avere lo stesso nome. Lo script che segue illustra un esempio del campo di opzione:

```
<form action='<?php echo $_SERVER["PHP_SELF"]; ?>' method='post'>

<h2>Seleziona il tuo colore preferito:</h2>
Blu <input type='radio' name='col' value='blu'>
Verde <input type='radio' name='col' value='verde'>
Giallo  <input type='radio' name='col' value='giallo'>
Rosso  <input type='radio' name='col' value='rosso'>
<br>
<h2>Seleziona il tipo di musica che preferisci:</h2>
Pop <input type='radio' name='mus' value='pop'>
Rock <input type='radio' name='mus' value='rock'>
Classica  <input type='radio' name='mus' value='classica'>
<br><br>
```

```
<input type='submit' name='submit'>

</form>

<?php

// Interazione con l'utente - Esempio 18-6
//-----------------------------------

if(isset($_POST["submit"]))
{
     $col = $_POST["col"];
     $mus = $_POST["mus"];
     echo("Il tuo colore preferito è il $col<br>");
     echo("La musica che preferisci è $mus");
}

?>
```

Lo script visualizza due gruppi di pulsanti di opzione, uno chiamato `"col"` l'altro `"mus"`.
Viene effettuato un controllo per stabilire se il pulsante di invio è stato premuto e, in caso affermativo, vengono visualizzati i valori memorizzati nelle variabili `$col` e `$mus`.
L'output ottenuto dall'esecuzione dello script precedente è mostrato nella Figura 18.7.

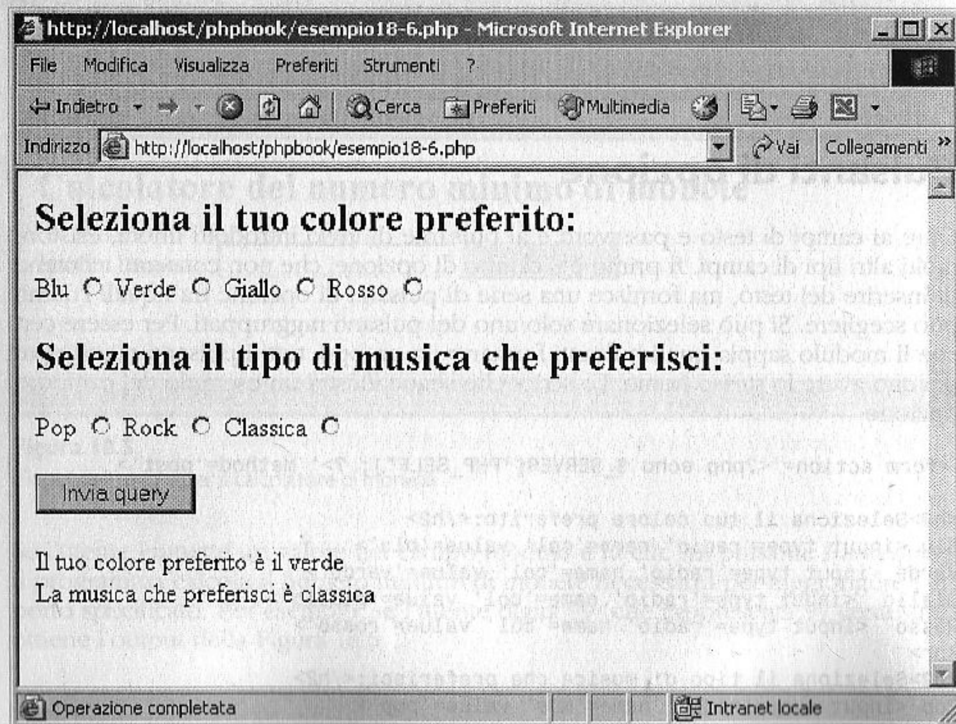


Figura 18.7
Pulsanti di opzione.

Campi caselle di controllo

I campi caselle di controllo permettono all'utente di selezionare tutte le opzioni che richiede al modulo. A ogni casella di controllo viene assegnato un nome univoco. Lo script che segue illustra un esempio dell'impiego del campo casella di controllo:

```
<html>
   <body>
      <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

      <h2>Seleziona tutti i colori che ti piacciono:</h2>
      Blu <input type='checkbox' name='cb1' value='blu'>
      Verde <input type='checkbox' name='cb2' value='verde'>
      Giallo  <input type='checkbox' name='cb3' value='giallo'>
      Rosso <input type='checkbox' name='cb4' value='rosso'>
      <br><br>
      <input type='submit' name='submit'>

      </form>
   </body>
</html>

<?php

// Interazione con l'utente - Esempio 18-7
//-----------------------------------

if(isset($_POST["submit"]))
{
   if (isset($_POST["cb1"]))
        $cb1 = $_POST["cb1"];
   else
        $cb1 = "";
   if (isset($_POST["cb2"]))
        $cb2 = $_POST["cb2"];
   else
        $cb2 = "";
   if (isset($_POST["cb3"]))
        $cb3 = $_POST["cb3"];
   else
        $cb3 = "";
   if (isset($_POST["cb4"]))
        $cb4 = $_POST["cb4"];
   else
        $cb4 = "";
   echo("I colori che ti piacciono sono $cb1 $cb2 $cb3 $cb4");
}

?>
```

Lo script visualizza un modulo composto da quattro caselle di controllo, che rappresentano colori diversi. All'utente viene chiesto di selezionare un colore qualsiasi. Se il pulsante di invio è stato premuto, vengono visualizzati i valori delle variabili delle caselle di controllo (`$cb1`, `$cb2`, `$cb3` e `$cb4`).
L'output ottenuto dall'esecuzione dello script precedente è quello della Figura 18.8

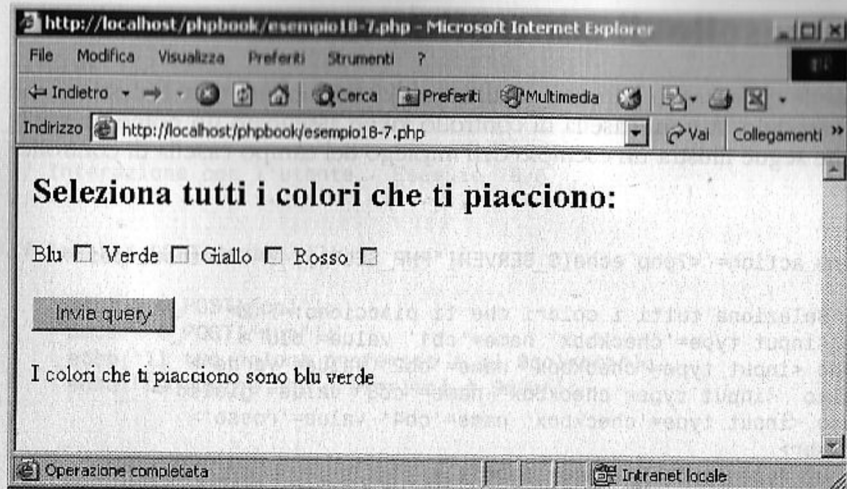


Figura 18.8
Campi caselle di controllo.

Campi di selezione

Il campo di selezione del modulo utilizza un formato diverso, rispetto agli altri campi modulo incontrati in precedenza. Esso utilizza due elementi HTML: `select` e `option`. Ogni volta che volete visualizzare il menu di selezione a discesa, dovete inserire un elemento `option` all'interno dell'elemento `select`. Dato che ogni elemento `option` comprende tutti i dati, non c'è alcun bisogno di inserire attributi per i valori. Nello script seguente è mostrato un esempio dell'utilizzo di questo elemento:

```
<form action='<?phpz echo($_SERVER["PHP_SELF"]) ?>' method='post'>
   Selezionare il proprio titolo: <select name='title'>
                                  <option>Sig.</option>
                                  <option>Sig.ra</option>
                                  <option>Dr.</option>
                                  <option>Ing.</option>
                                  <option>Prof.</option>
                                  </select>
   <br><br>
   <input type='submit' name='submit'>
</form>

<?php

// Interazione con l'utente - Esempio 18-8
//----------------------------------

if(isset($_POST["submit"])){
    $title = $_POST["title"];
    echo("Il titolo scelto è $title");
}

?>
```

Lo script visualizza un modulo composto da un solo menu di selezione, di nome `title`, costituito da una selezione di titoli, e verifica se il pulsante di invio è stato premuto e, in caso affermativo, visualizza il valore `$title`. L'output ottenuto dall'esecuzione dello script precedente è quello della Figura 18.9.

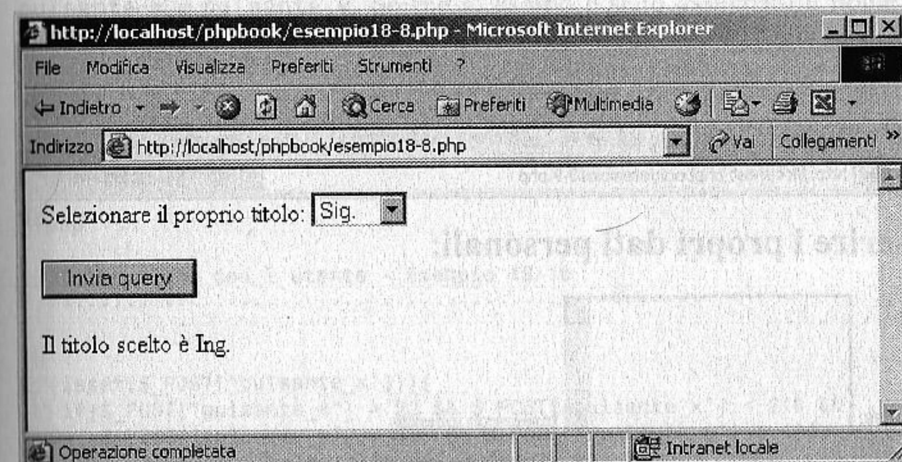


Figura 18.9
Campi di selezione.

Campi textarea

L'elemento `textarea` (ossia area di testo) del modulo consente di creare una casella di testo composta da un certo numero di righe e colonne. Considerate lo script seguente:

```
<?php

// Interazione con l'utente - Esempio 18-9
//----------------------------------

if(isset($_POST["submit"])){
    if($_POST["address"] == "")
        $addressErr = 1;
}

?>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
   <h2>Inserire i propri dati personali:</h2>
   Indirizzo: <textarea name='address' rows='5'></textarea>
              <?php
                 if(isset($addressErr))
                    echo("<font color='red' size='2'> Inserire un
                          indirizzo</font>");
              ?>
   <br><br>
   <input type='submit' name='submit'>
</form>
```

Lo script visualizza un modulo costituito da un campo **textarea**. All'utente viene chiesto di inserire il proprio indirizzo nel campo. Se il pulsante di invio viene premuto senza che nell'area di testo sia stato inserito qualcosa, compare un messaggio che invita l'utente a inserire un indirizzo. La Figura 18.10 mostra l'output di questo script.



Figura 18.10
Campi textarea.

Campi immagine

I moduli possono anche elaborare le immagini come campi di input. Anche se questa affermazione di primo acchito può sembrare strana, provate a considerare l'immagine illustrata nella Figura 18.11, che è composta da tre pulsanti.

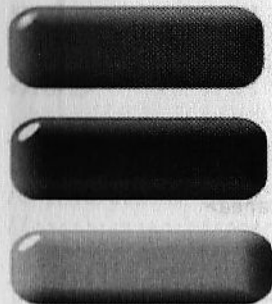


Figura 18.11
Immagine dei pulsanti.

Il campo immagine del modulo è rappresentato nello script seguente. Il programma non comprende un pulsante di invio, poiché quando l'utente fa clic sull'immagine, le coordinate x e y del punto in cui l'utente ha fatto clic vengono trasmesse allo script.

Questi valori sono memorizzati in due variabili, denominate rispettivamente `pulsante_x` e `pulsante_y`, poiché al campo è stato assegnato il nome `pulsante`:

```
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
    Fare clic sull'immagine:<br>
    <input type='image' src='graphics/pulsanti.jpg' name='pulsante'>
    </form>

<?php

// Interazione con l'utente - Esempio 18-10
//----------------------------------------------------


if(isset($_POST["pulsante_x"])){
   if($_POST["pulsante_x"] > 22 && $_POST["pulsante_x"] < 210 &&
      $_POST["pulsante_y"] < 155 && $_POST["pulsante_y"] > 100)
         echo("Hai fatto clic sul pulsante blu.");
   else
         echo("Hai fatto clic sull'immagine.");
}

?>
```

Lo script precedente visualizza l'immagine come campo immagine del modulo. Dato che manca un pulsante di invio, l'unica azione in grado di inviare il modulo è il clic dell'utente sull'immagine.

Quando l'utente fa clic sull'immagine, lo script verifica i valori X e Y passati come variabili `pulsante_x` e `pulsante_y` per stabilire se l'utente ha fatto clic sul pulsante blu (quello in mezzo) e visualizza un messaggio di conferma, come mostrato nella Figura 18.12. È possibile stabilire se l'utente ha fatto clic sul pulsante centrale, poiché è noto che l'angolo in alto a sinistra del pulsante si trova nella posizione 22 × 100, mentre l'angolo in basso a destra occupa la posizione 210 × 155 dell'immagine.

Se i valori memorizzati in `pulsante_x` e `pulsante_y` sono compresi in questi intervalli, significa che l'utente ha fatto clic sul pulsante in questione.

Caricamenti di file

I moduli possono utilizzare un elemento che permette all'utente di selezionare un file dal computer locale e di caricarlo nel server, in modo che lo script PHP possa accedervi. Si tratta di un processo complesso, analizzato in dettaglio nel Capitolo 28.

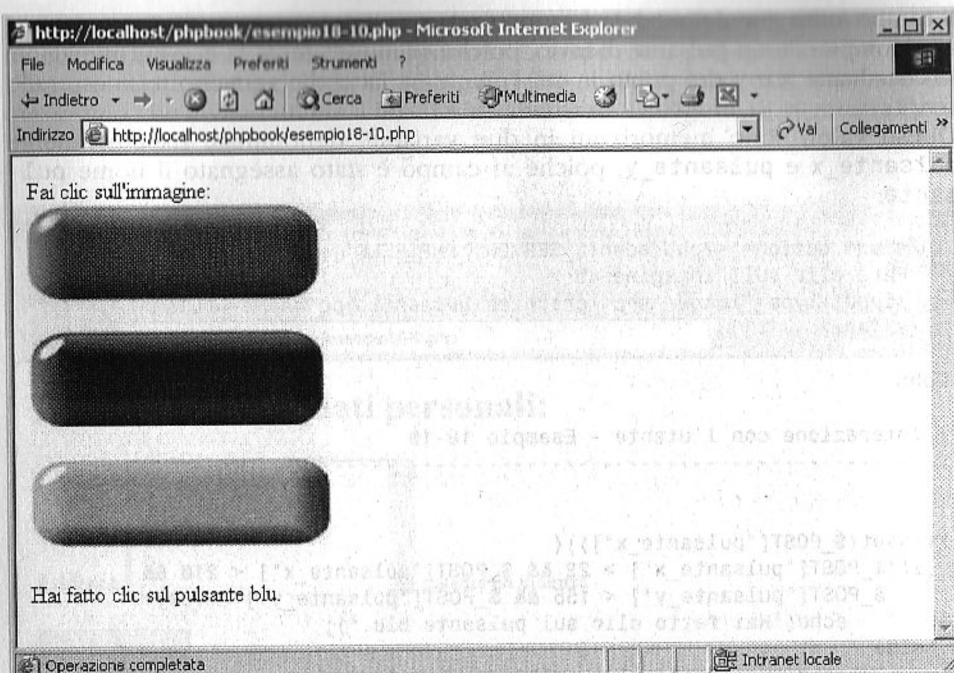


Figura 18.12
Output del campo immagine.

Riepilogo

Questo capitolo ha spiegato come utilizzare PHP per interagire con l'utente attraverso i moduli HTML. Sono stati presentati i diversi elementi dei moduli e sono stati illustrati i metodi per estrarre i dati contenuti negli elementi. Tuttavia, l'interazione con l'utente non si limita alla possibilità di visualizzare un modulo e accedere ai dati in esso contenuti. Il prossimo capitolo affronterà il ruolo della convalida dei dati dei moduli e della rivisualizzazione dei dati dei moduli.

Capitolo 19

Convalida dei moduli e conservazione dei dati

Introduzione

L'interazione con l'utente è efficace se i dati che egli ha inserito sono effettivamente quelli richiesti. Come è noto, può accadere che gli utenti digitino informazioni non richieste, pertanto la convalida dei dati dei moduli è fondamentale. Provate a immaginare che cosa accadrebbe se, tra i dati inseriti nel modulo, si riscontrasse un errore: dovreste dare all'utente la possibilità di visualizzare nuovamente il modulo, in modo da poter correggere l'errore.
Tuttavia sarebbe una grande perdita di tempo se l'utente fosse costretto a inserire nuovamente tutti i dati del modulo, compresi quelli corretti. È necessario pertanto uno strumento che consenta di conservare e rivisualizzare i dati dei moduli. Questo capitolo spiega come convalidare e come rivisualizzare i dati dei moduli.

Convalida di presenza o assenza

Una delle forme più semplici di convalida consiste nello stabilire se l'utente ha inserito o meno un qualche dato in un campo del modulo. Controllare che un campo di testo o password contenga informazione equivale a verificare se la variabile creata dal campo modulo ha lunghezza zero. A tal fine si può utilizzare un'istruzione `if` come la seguente:

```
if ($variable == "")
```

Nel caso di un campo di selezione, le cose sono leggermente diverse; infatti, di default, la prima voce del menu è selezionata automaticamente. Questa impostazione non crea alcun problema se questa voce è ciò che l'utente vuole, ma se l'utente non intende selezionare tale elemento e si è persino dimenticato di inserirlo, la situazione si complica; per risolverla è possibile inserire una voce di menu "falsa" all'inizio del menu stesso. Successivamente potete controllare il valore della variabile di questa voce e scartarla qualora venisse trovata.

Lo script che segue illustra la convalida di un elemento di selezione del modulo.

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-1
//-----------------------------------

if(isset($_POST["submit"])){
   if($_POST["title"] == "Selezionare...")
      $titErr = 1;
   else
      $titErr = 0;
   if($_POST["firstname"] == "")
      $firstErr = 1;
   else
      $firstErr = 0;
   if($_POST["surname"] == "")
      $surErr = 1;
   else
      $surErr = 0;
   if (isset($_POST["mus"]))
      {
        if ($_POST["mus"] == "")
           $musErr = 1;
        else
           $musErr = 0;
      }
   else
     $musErr = 1;
}
else {
  $titErr = 0;
  $firstErr = 0;
  $surErr = 0;
  $musErr = 0;
}

?>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i propri dati personali:</h2>
Titolo: <select name='title'>
        <option>Selezionare...</option>
        <option>Sig.</option>
        <option>Sig.ra</option>
        <option>Dr.</option>
        <option>Ing.</option>
        <option>Prof.</option>
        </select>
        <?php
         if($titErr)
           echo("<font color='red' size='2'> Selezionare untitolo</font>");
        ?>

<br>
```



```
Nome: <input type='text' name='firstname'>
    <?php
     if($firstErr)
       echo("<font color='red' size='2'> Inserire il nome</font>");
    ?>

<br>
Cognome: <input type='text' name='surname'>
         <?php
          if($surErr)
            echo("<font color='red' size='2'> Inserire il cognome</font>");
         ?>

<br>
<h2>Selezionare il genere musicale preferito:</h2>
Pop <input type='radio' name='mus' value='pop'>
Rock <input type='radio' name='mus' value='rock'>
Classica  <input type='radio' name='mus' value='classica'>
          <?php
           if($musErr)
              echo("<font color='red' size='2'> Selezionare il genere
                   musicale preferito</font>");
          ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

Lo script inizia stabilendo se il modulo è stato inviato. In caso affermativo, il valore `title`, una voce del modulo di selezione, viene controllato per vedere se equivale al valore "Seleziona... ". Si tratta della prima opzione di menu e non è un input valido. Le tre istruzioni `if` successive verificano se i valori `firstname`, `surname` e `mus` sono vuoti. Se qualche variabile del modulo è vuota, le variabili di errore corrispondenti vengono impostate a 1.

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-1
//-----------------------------------

if(isset($_POST["submit"])){
        if($_POST["title"] == "Selezionare...")
              $titErr = 1;
        else
              $titErr = 0;
        if($_POST["firstname"] == "")
              $firstErr = 1;
        else
              $firstErr = 0;
        if($_POST["surname"] == "")
              $surErr = 1;
        else
              $surErr = 0;
        if (isset($_POST["mus"]))
           {
              if ($_POST["mus"] == "")
```

```
            $musErr = 1;
        else
            $musErr = 0;
    }
    else
        $musErr = 1;
}
else {
    $titErr = 0;
    $firstErr = 0;
    $surErr = 0;
    $musErr = 0;
}
?>
```

La parte successiva dello script visualizza il modulo e il suo primo elemento, il campo di selezione:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i propri dati personali:</h2>
Titolo: <select name='title'>
        <option>Selezionare...</option>
        <option>Sig.</option>
        <option>Sig.ra</option>
        <option>Dr.</option>
        <option>Ing.</option>
        <option>Prof.</option>
        </select>
```

La parte successiva controlla il valore della variabile `$titErr` (la variabile impostata nel caso in cui si rilevi un errore), e se tale valore è 1 visualizza un messaggio di errore:

```
        <?php
         if($titErr)
           echo("<font color='red' size='2'> Selezionare untitolo</font>");
        ?>
```

La parte finale dello script visualizza gli altri tre campi del modulo e controlla le variabili di errore corrispondenti (`$firstErr`, `$surErr` e `$musErr`) per decidere se visualizzare o meno un messaggio di errore.

```
<br>
Nome: <input type='text' name='firstname'>
      <?php
       if($firstErr)
          echo("<font color='red' size='2'> Inserire il nome</font>");
      ?>

<br>
Cognome: <input type='text' name='surname'>
         <?php
          if($surErr)
            echo("<font color='red' size='2'> Inserire il cognome</font>");
         ?>
```



```
<br>
<h2>Selezionare il genere musicale preferito:</h2>
Pop <input type='radio' name='mus' value='pop'>
Rock <input type='radio' name='mus' value='rock'>
Classica  <input type='radio' name='mus' value='classica'>
          <?php
           if($musErr)
               echo("<font color='red' size='2'> Selezionare il genere
                    musicale preferito</font>");
          ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

La Figura 19.1 illustra l'output prodotto da questo script, in cui un utente si è dimenticato di inserire qualche dato nel modulo.

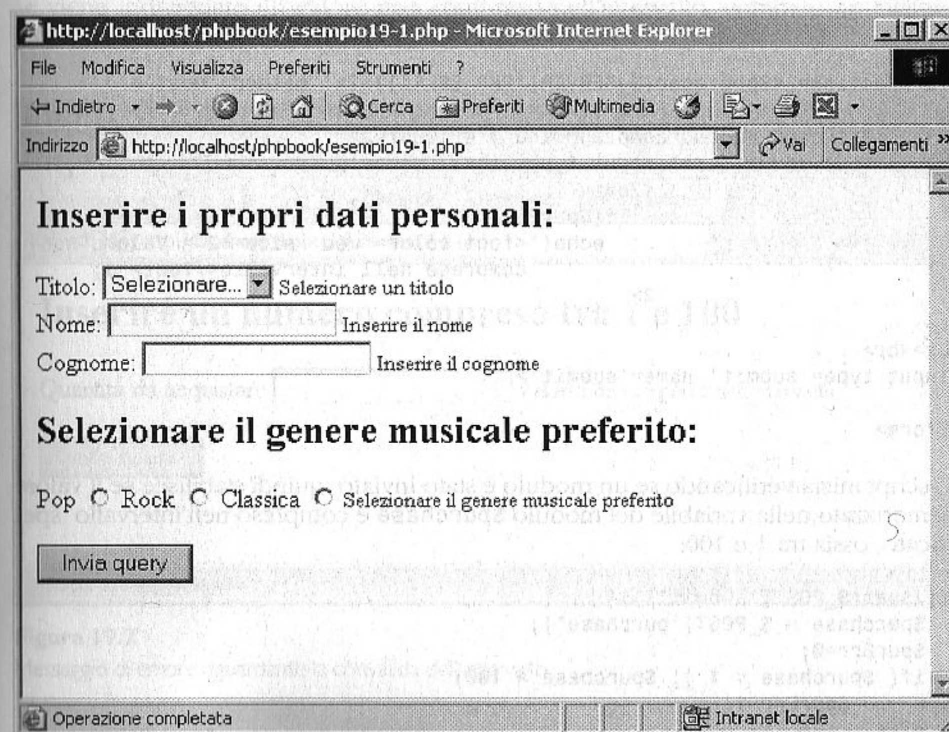


Figura 19.1
Messaggi di errore per i dati dei moduli presenti o assenti.

Convalida dell'intervallo

Anche la convalida dell'intervallo è un'operazione piuttosto semplice. Supponete di chiedere all'utente di inserire un valore compreso tra due numeri, per esempio il

numero di prodotti che ha intenzione di acquistare. A tal fine potete utilizzare una semplice istruzione if. Lo script seguente illustra come.

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-2
//-------------------------------

if(isset($_POST["submit"])){
   $purchase = $_POST["purchase"];
   $purErr=0;
   if( $purchase < 1 || $purchase > 100)
         $purErr=1;
   else
         echo("Hai scelto di acquistare $purchase prodotti.");
}
else
   $purErr=0;

?>

<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire un numero compreso tra 1 e 100</h2>
Quantità da acquistare:<input type='text' name='purchase'>
                              <?php
                                if($purErr)
                                  echo("<font color='red' size='2'> Valore non
                                        compreso nell'intervallo</font>");
                              ?>

<br><br>
<input type='submit' name='submit'>

</form>
```

Lo script inizia verificando se un modulo è stato inviato, quindi stabilisce se il valore memorizzato nella variabile del modulo $purchase è compreso nell'intervallo specificato, ossia tra 1 e 100:

```
if(isset($_POST["submit"])){
   $purchase = $_POST["purchase"];
   $purErr=0;
   if( $purchase < 1 || $purchase > 100)
         $purErr=1;
   else
         echo("Hai scelto di acquistare $purchase prodotti.");
}
else
   $purErr=0;
```

Se il valore rientra nell'intervallo stabilito, compare un messaggio con il numero dei prodotti che si desiderano acquistare. In caso contrario la variabile $purErr viene impostata a 1. Poi viene visualizzato nuovamente il modulo con un singolo campo

di testo e la richiesta di inserire un valore compreso tra 1 e 100. Se invece il valore di $purErr è 1, compare un messaggio di errore:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire,un numero compreso tra 1 e 100</h2>
Quantità da acquistare:<input type='text' name='purchase'>
                              <?php
                                if($purErr)
                                  echo("<font color='red' size='2'> Valore non
                                        compreso nell'intervallo</font>");
                              ?>

<br><br>
<input type='submit' name='submit'>

</form>
```

Se viene individuato un valore non compreso nell'intervallo, compare un messaggio di errore, come quello della Figura 19.2.

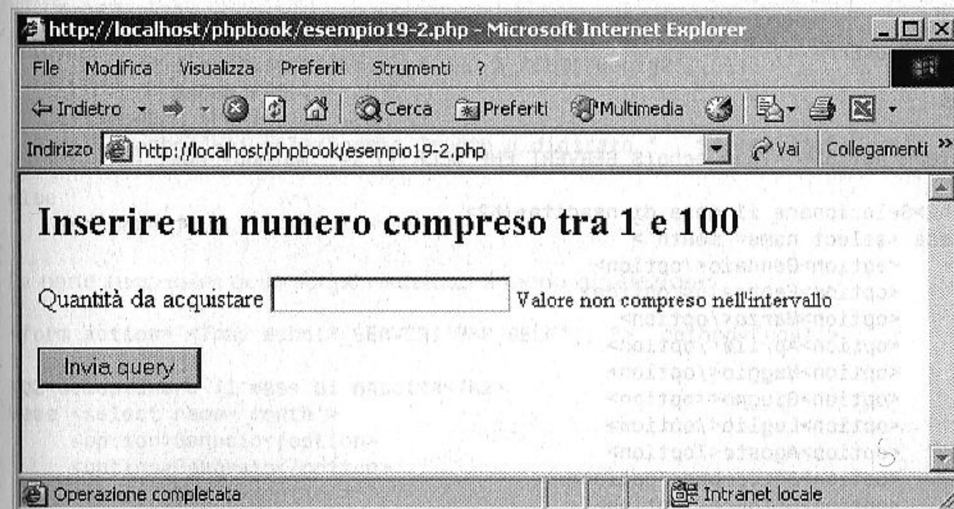


Figura 19.2
Messaggio di errore riguardante la convalida dell'intervallo.

Convalida delle stringhe

Accade spesso di commettere errori nella digitazione del testo, anche nella compilazione di semplici moduli. Una soluzione consiste nel ridurre al minimo la quantità di testo che l'utente deve digitare. Per ottenere questo risultato, si potrebbe fornire il maggior numero possibile di voci del modulo attraverso menu di selezione, eliminando il problema alla radice. Tuttavia, non sempre è possibile ricorrere a questa soluzione. Lo script che segue mostra due metodi per ottenere e convalidare una

stringa di input che rappresenta un mese dell'anno. Il modulo contiene un menu di selezione da cui l'utente può scegliere un mese (il metodo semplice), e un campo per l'immissione di testo in cui l'utente deve inserire la stringa del mese. È questo il campo che dobbiamo convalidare.

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-3
//-------------------------------

$months = array("Gennaio","Febbraio","Marzo","Aprile","Maggio","Giugno",
"Luglio","Agosto","Settembre","Ottobre","Novembre","Dicembre");

if(isset($_POST["submit"])){
    $month = $_POST["month"];
    $monErr=-1;
    for($c=0;$c<12;$c++)
        if(!strncasecmp($months[$c],$_POST["month2"],3))
            $monErr=$c;
    if($monErr != -1)
        echo("Hai selezionato $month e digitato " . $months[$monErr]);
}
else
    $monErr=-1;

?>

<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Selezionare il mese di nascita</h2>
Mese <select name='month'>
    <option>Gennaio</option>
    <option>Febbraio</option>
    <option>Marzo</option>
    <option>Aprile</option>
    <option>Maggio</option>
    <option>Giugno</option>
    <option>Luglio</option>
    <option>Agosto</option>
    <option>Settembre</option>
    <option>Ottobre</option>
    <option>Novembre</option>
    <option>Dicembre</option>
    </select>
<br>
Mese <input type='text' name='month2'>
    <?php
      if($monErr == -1)
        echo("<font color='red' size='2'> Mese non riconosciuto</font>");
    ?>
<br><br>
<input type='submit' name='submit'>

</form>
```


Lo script inizia dichiarando un array `$months` e lo popola con i mesi dell'anno. Successivamente un'istruzione `if` controlla se il modulo è stato inviato:

```
$months = array("Gennaio","Febbraio","Marzo","Aprile","Maggio","Giugno",
"Luglio","Agosto","Settembre","Ottobre","Novembre","Dicembre");

if(isset($_POST["submit"])){
```

Quindi una variabile `$monErr` viene dichiarata e impostata a –1. Questa variabile servirà per memorizzare la posizione del mese nell'array, qualora venga trovato. Un ciclo `for` viene iterato 12 volte per mettere a confronto ciascun mese presente nell'array con i dati inseriti nel modulo. A tal fine si utilizza la funzione `strncasecmp()`, introdotta nel Capitolo 17, che confronta i primi tre caratteri dei dati immessi con i primi tre dell'elemento dell'array corrente. La funzione non distingue tra minuscole e maiuscole, quindi non ha alcuna importanza se l'utente ha digitato il nome del mese con un carattere maiuscolo o minuscolo.
Se la funzione trova una corrispondenza, la variabile `$monErr` viene impostata al valore corrente del conteggio dell'array. Se dopo il confronto di tutti i dodici elementi nell'array si trova una corrispondenza, compare un messaggio:

```
    $month = $_POST["month"];
    $monErr=-1;
    for($c=0;$c<12;$c++)
        if(!strncasecmp($months[$c],$_POST["month2"],3))
            $monErr=$c;
    if($monErr != -1)
        echo("Hai selezionato $month e digitato " . $months[$monErr]);
}
else
    $monErr=-1;
```

La parte successiva dello script visualizza il menu di selezione:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Selezionare il mese di nascita</h2>
Mese <select name='month'>
    <option>Gennaio</option>
    <option>Febbraio</option>
    <option>Marzo</option>
    <option>Aprile</option>
    <option>Maggio</option>
    <option>Giugno</option>
    <option>Luglio</option>
    <option>Agosto</option>
    <option>Settembre</option>
    <option>Ottobre</option>
    <option>Novembre</option>
    <option>Dicembre</option>
    </select>
<br>
```

Infine, se il flag `$monErr` è stato impostato, viene visualizzato il campo testo del modulo insieme a un messaggio di errore:

```
Mese <input type='text' name='month2'>
```

```
    <?php
      if($monErr == -1)
          echo("<font color='red' size='2'> Mese non riconosciuto</font>");
    ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

La Figura 19.3 illustra l'output relativo al messaggio d'errore dello script precedente.

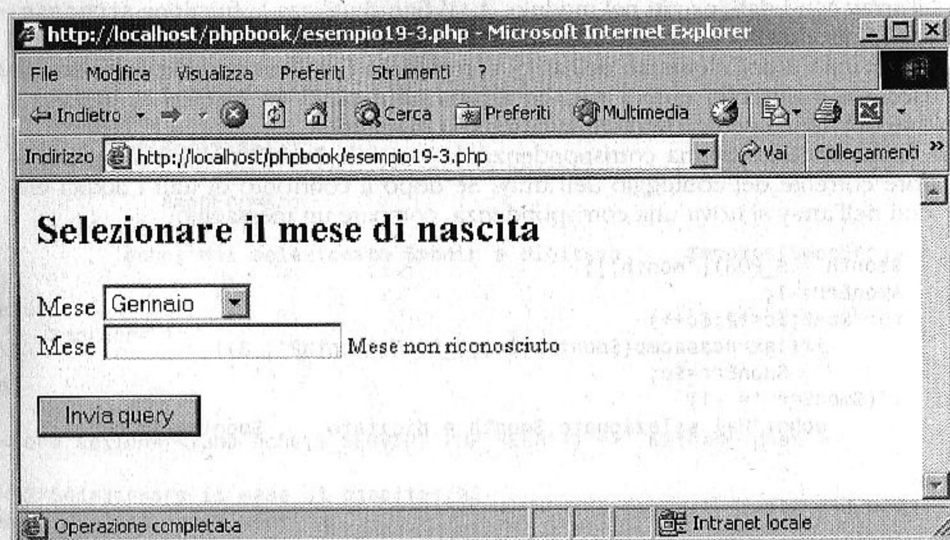


Figura 19.3
Errore di mese non riconosciuto.

Conservazione dei dati nei campi testo e password

Negli esempi di convalida dei moduli mostrati finora, qualunque dato inserito dall'utente nel modulo andava perduto quando il modulo veniva nuovamente visualizzato insieme ai corrispondenti messaggi di errore. La scelta di non visualizzare nuovamente i dati considerati errati potrebbe costituire una buona soluzione, ma non inserire i campi che hanno superato il test di convalida è molto frustrante per l'utente, che si ritrova costretto a digitare di nuovo le informazioni; nella migliore delle ipotesi è una semplice seccatura, che però potrebbe portare l'utente a commettere nuovi errori nell'immissione dei dati. In questo caso, la soluzione migliore sarebbe visualizzare i dati inseriti in modo corretto nei campi del modulo quando quest'ultimo viene riproposto all'utente.

Lo script che segue mostra un esempio di rivisualizzazione dei dati del modulo all'interno di un campo testo:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-4
//-----------------------------------

if(isset($_POST["submit"])){
   $surname = $_POST["surname"];
   $firstname = $_POST["firstname"];

   if($firstname == "")
      $firstErr = 1;
   else
      $firstErr = 0;
   if($surname == "")
      $surErr = 1;
   else
      $surErr = 0;
}
else {
      $firstErr = 0;
      $surErr = 0;
      $surname = "";
      $firstname = "";
}
?>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i dati personali:</h2>
Nome: <input type='text' name='firstname'
        value='<?php echo($firstname)?>'>
      <?php
        if($firstErr)
           echo("<font color='red' size='2'> Inserire il nome</font>");
      ?>
<br/>
Cognome:<input type='text' name='surname'
         value='<?php echo($surname)?>'>
       <?php
         if($surErr)
            echo("<font color='red' size='2'> Inserire il cognome</font>");
       ?>
<br /><br />
<input type='submit' name='submit'>

</form>
```

Lo script inizia controllando se il modulo è stato inviato e, in caso affermativo, verifica se nei campi `firstname` e `surname` sono stati inseriti dei dati:

```
if(isset($_POST["submit"])){
```

```
        $surname = $_POST["surname"];
        $firstname = $_POST["firstname"];

        if($firstname == "")
                $firstErr = 1;
        else
                $firstErr = 0;
        if($surname == "")
                $surErr = 1;
        else
                $surErr = 0;
}
else {
        $firstErr = 0;
        $surErr = 0;
        $surname = "";
        $firstname = "";
}
?>
```

La parte successiva dello script visualizza l'inizio del modulo:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i dati personali:</h2>
```

Di seguito è riportata quella parte "intelligente" del codice che permette di conservare i dati contenuti nel modulo. Osservate che i campi di testo `firstname` e `surname` hanno un attributo `value`. Osservate inoltre lo script PHP che è stato aggiunto per riprodurre il valore delle variabili `$firstname` e `$surname` nell'attributo `value`. In questo modo è certo che i valori contenuti in queste variabili compariranno nel campo di testo quando il modulo sarà visualizzato, conservando così i dati:

```
Nome: <input type='text' name='firstname'
       value='<?php echo($firstname)?>'>
      <?php
        if($firstErr)
           echo("<font color='red' size='2'> Inserire il nome</font>");
      ?>
<br/>
Cognome:<input type='text' name='surname'
         value='<?php echo($surname)?>'>
        <?php
         if($surErr)
            echo("<font coлlor='red' size='2'> Inserire il cognome</font>");
        ?>
<br /><br />
<input type='submit' name='submit'>

</form>
```

La Figura 19.4 mostra un modulo inviato. Mentre il nome è stato inserito, il cognome manca, e ciò comporta la visualizzazione di un messaggio di errore. Osservate che il dato `firstname` compare nuovamente nel modulo.

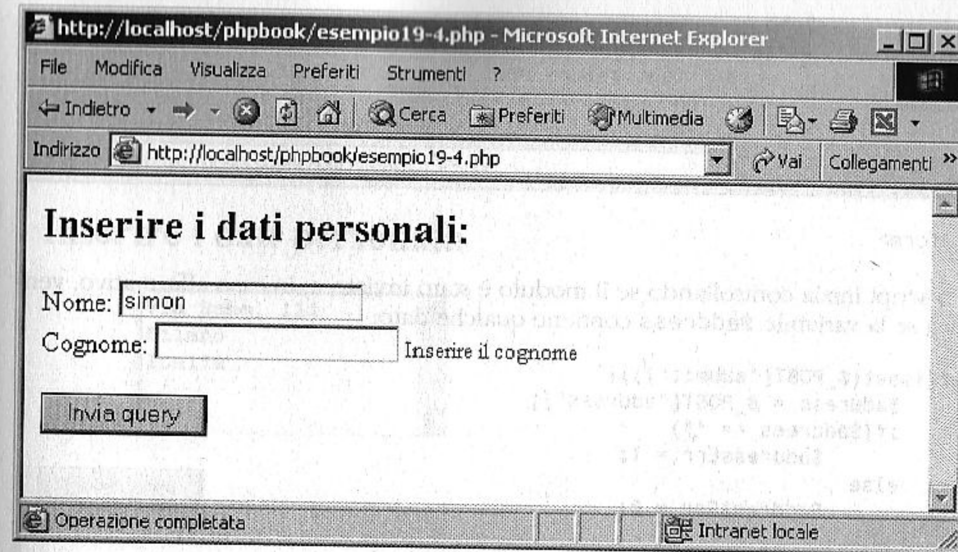


Figura 19.4
Conservazione dei dati in un campo di testo.

Conservazione dei dati nei campi textarea

La conservazione dei dati inseriti in un campo textarea è facile. Considerate lo script seguente:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-5
//----------------------------------

if(isset($_POST["submit"])){
    $address = $_POST["address"];
    if($address == "")
          $addressErr = 1;
    else
          $addressErr = 0;
}
else {
    $addressErr = 1;
    $address = "";
}

?>

<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i dati personali:</h2>
Indirizzo: <textarea name='address' rows='5'>
           <?php echo($address) ?>
           </textarea>
```

```
        <?php
          if($addressErr)
            echo("<font color='red' size='2'> Inserire un
                   indirizzo</font>");
        ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

Lo script inizia controllando se il modulo è stato inviato e, in caso affermativo, verifica se la variabile `$address` contiene qualche dato:

```
if(isset($_POST["submit"])){
     $address = $_POST["address"];
     if($address == "")
           $addressErr = 1;
     else
           $addressErr = 0;
}
else {
     $addressErr = 1;
     $address = "";
}
?>
```

La parte successiva dello script visualizza il modulo:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Inserire i dati personali:</h2>
```

Per conservare i dati immessi nel campo modulo area di testo, il valore della variabile `$address` viene visualizzato fra i tag iniziale e finale dell'elemento `textarea`:

```
Indirizzo: <textarea name='address' rows='5'>
           <?php echo($address) ?>
           </textarea>
           <?php
             if($addressErr)
               echo("<font color='red' size='2'> Inserire un
                      indirizzo</font>");
           ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

L'output ottenuto dallo script precedente è quello della Figura 19.5.

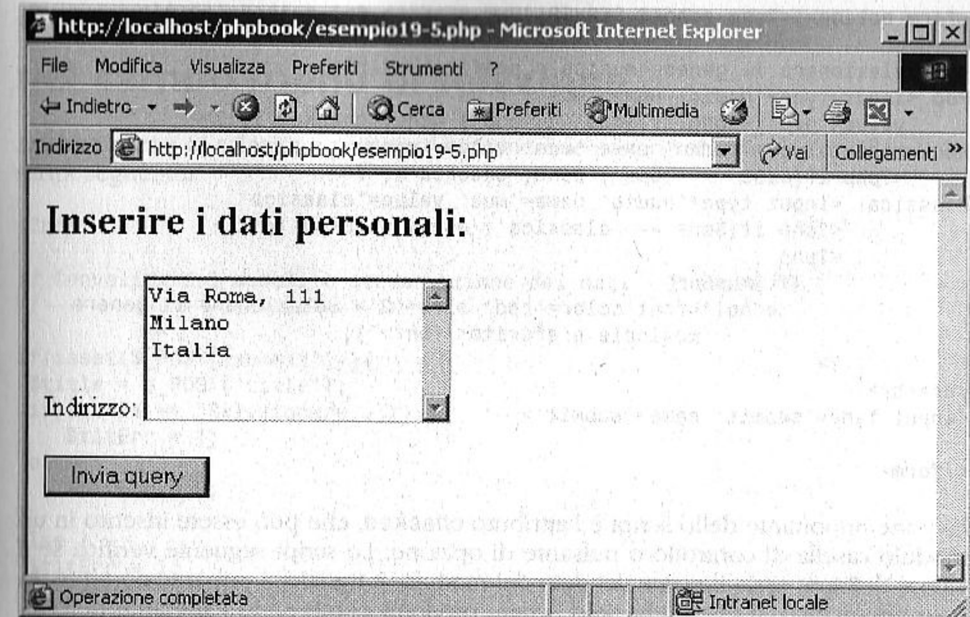


Figura 19.5
Conservazione dei dati nel campo textarea.

Conservazione dei dati nei campi casella di controllo e pulsante di opzione

Gli elementi casella di controllo e pulsante di opzione possono conservare le proprie selezioni più o meno come i campi testo e password. Lo script che segue illustra come viene mantenuta la selezione effettuata dall'utente:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-6
//-----------------------------------

if(isset($_POST["submit"])){
   $mus = $_POST["mus"];
   if($mus == "")
      $musErr = 1;
   else
      $musErr = 0;
}
else {
   $musErr = 1;
   $mus = "";
}

?>
```

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Selezionare il genere musicale preferito:</h2>
Pop <input type='radio' name='mus'
    value='pop' <?php if($mus == 'pop') echo('checked'); ?>>
Rock <input type='radio' name='mus' value='rock'
    <?php if($mus == 'rock') echo('checked'); ?>>
Classica  <input type='radio' name='mus' value='classica'
          <?php if($mus == 'classica') echo('checked'); ?>>
          <?php
            if($musErr)
              echo("<font color='red' size='2'> Selezionare il genere
                    musicale preferito</font>");
          ?>
<br><br>
<input type='submit' name='submit'>

</form>
```

La parte importante dello script è l'attributo **checked**, che può essere inserito in un modulo casella di controllo o pulsante di opzione. Lo script seguente verifica se il valore di **$mus**, ossia il nome del dato del modulo, è uguale al pulsante corrente. In caso affermativo, il testo "checked" viene prodotto come parte integrante dell'elemento del modulo.

La Figura 19.6 illustra l'output ottenuto dallo script precedente. Osservate che, al momento dell'invio del modulo, la selezione "Classica" dell'utente è stata mantenuta.

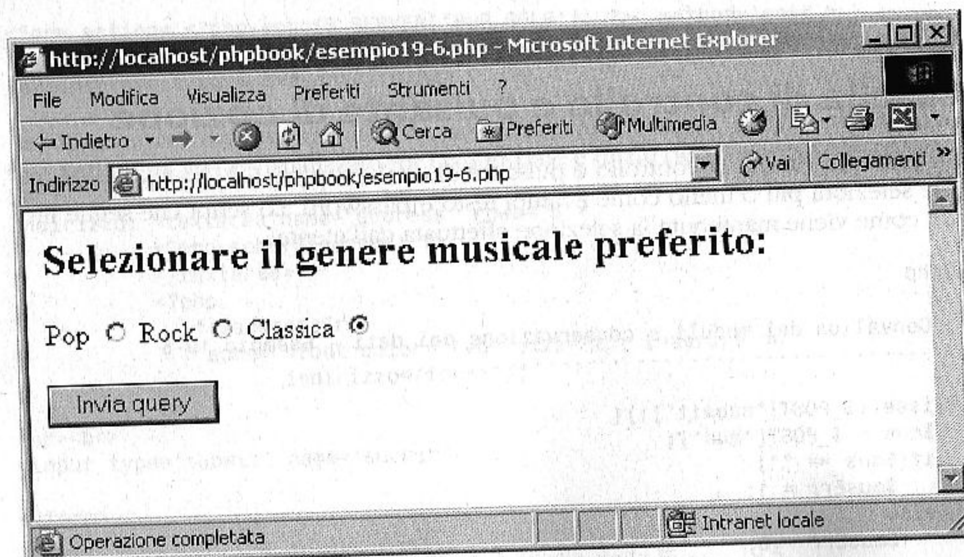


Figura 19.6
Conservazione del pulsante di opzione selezionato.

Conservazione dei dati nel campo menu di selezione

Anche i menu di selezione possono mantenere le proprie selezioni. Considerate lo script seguente:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-7
//-----------------------------------

if(isset($_POST["submit"])){
 $title = $_POST["title"];
 if($title == "Selezionare...")
    $titErr = 1;
 else
    $titErr = 0;
}
else {
 $titErr = 1;
 $title = "";
}

?>

<body>
  <html>
    <form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

    <h2>Inserire i dati personali:</h2>
    Titolo: <select name='title'>

            <option>Selezionare...</option>
            <option <?php if($title == "Sig.") echo("Selected")?>>
              Sig.</option>
            <option <?php if($title == "Sig.ra") echo("Selected")?>>
              Sig.ra</option>
            <option <?php if($title == "Dr.") echo("Selected") ?>>
              Dr.</option>
            <option <?php if($title == "Ing.") echo("Selected") ?>>
              Ing.</option>
            <option <?php if($title == "Prof.") echo("Selected") ?>>
              Prof.</option>

            </select>
            <?php
              if($titErr)
                echo("<font color='red' size='2'> Selezionare un
                      titolo</font>");
            ?>
     <br><br>
     <input type='submit' name='submit'>

     </form>
  </body>
</html>
```

Lo script visualizza le opzioni relative alle selezioni del menu e controlla se il valore della variabile `$title`, che memorizza il valore dell'opzione selezionata, è uguale all'opzione corrente. In caso affermativo, l'attributo "selected" viene prodotto come parte dell'elemento `option`. La Figura 19.7 illustra che l'opzione "Sig." del menu di selezione inviato è stata mantenuta.

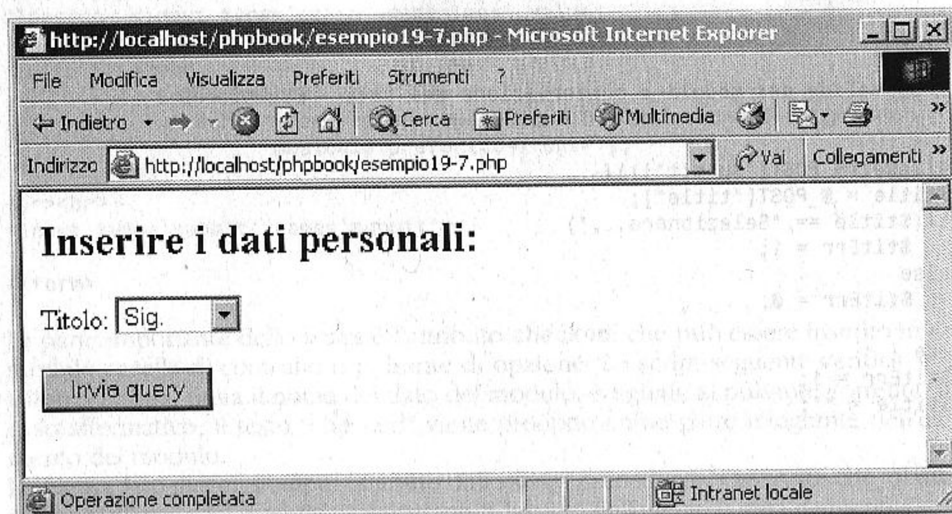


Figura 19.7
Conservazione di un'opzione scelta tra gli elementi di selezione.

Dati nascosti

I moduli possono essere utilizzati per passare dati ogni volta che un modulo viene visualizzato, senza che l'utente possa tuttavia vederli. A tal fine è possibile utilizzare l'elemento del modulo `hidden`.
Questa possibilità può rivelarsi utile per passare informazioni da uno script all'altro o restituire dati allo stesso script mediante un modulo, senza che l'utente veda i dati e possa interferire.

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-8
//---------------------------------

if(isset($_POST["submit"])){
      $count = $_POST["count"];
      $count++;
}
else
      $count = 1;

?>
```



```
<h2>Questo modulo è stato inviato <?php echo($count) ?> volte</h2>
<input type='hidden' name='count' value='<?php echo($count) ?>'>
<input type='submit' name='submit'>

</form>
```

Lo script è molto semplice: inizia controllando se il modulo è stato inviato e, in caso affermativo, incrementa il valore della variabile `$count`. Se il modulo non è stato inviato, il valore della variabile `$count` è impostato a 1:

```
if(isset($_POST["submit"])){
   $count = $_POST["count"];
   $count++;
}
else
   $count = 1;
?>
```

La parte successiva dello script visualizza il modulo e un titolo, che mostra il valore della variabile `$count`:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo è stato inviato <?php echo($count) ?> volte</h2>
```

Nascosto nel modulo c'è un nuovo tipo di elemento, noto con il nome di `hidden`. Gli elementi di tipo `hidden` possono accettare molti attributi utilizzati anche dagli altri elementi, come `name` e `value`.
In questo caso il nome dell'elemento è `count` (il nome della variabile), mentre il valore è il valore corrente della variabile `$count`:

```
<input type='hidden' name='count' value='<?php echo($count) ?>'>
<input type='submit' name='submit'>

</form>
```

In pratica, quando l'utente fa clic sul pulsante di invio, il valore di `$count` viene passato di nuovo allo script (anche se l'utente non si accorge di nulla), viene incrementato e visualizzato di nuovo. L'output ottenuto da questo script è quello della Figura 19.8.

Array e moduli

Finora, in questo capitolo, si è visto come i dati dei moduli possano essere conservati e/o nascosti all'interno del modulo. Un aspetto che non è stato analizzato è il passaggio degli array attraverso i moduli. Una delle ragioni di questa omissione è che gli array devono essere trattati in modo leggermente diverso rispetto alle altre variabili. Potreste pensare che è possibile includere un array in un modulo sotto forma di dato nascosto; tuttavia, come vedremo più avanti, ciò non è possibile. Lo

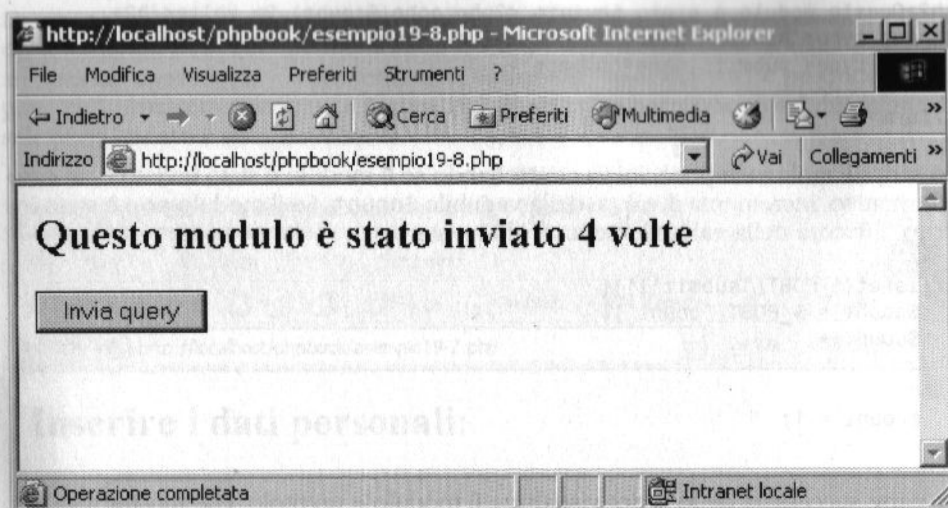


Figura 19.8
I dati nascosti del modulo permettono di contare quante volte il modulo è stato inviato.

script che segue verifica se il modulo è stato inviato; in caso affermativo, compare il contenuto di un array nascosto all'interno del modulo:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-9
//------------------------------------

if(isset($_POST["submit"])){
        echo("L'array contiene:<br>");
        $names = $_POST['names'];
        for($a=0;$a<4;$a++)
            echo($names[$a] . "<br>");
}
else
        $names = array("Simon","Liz","Gemma","Hayley");

?>

<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo contiene un array nascosto.</h2>
<input type='hidden' name='names' value='<?php echo($names) ?>'>
<input type='submit' name='submit'>

</form>
```

Viene utilizzato un ciclo `for` per visualizzare il contenuto dell'array:

```
if(isset($_POST["submit"])){
   echo("L'array contiene:<br>");
   $names = $_POST['names'];
   for($a=0;$a<4;$a++)
```



```
      echo($names[$a] . "<br>");
}
```

Se il modulo non è stato ancora inviato, l'array viene creato per la prima volta:

```
else
   $names = array("Simon","Liz","Gemma","Hayley");

?>
```

Il modulo viene visualizzato e, come spiegato in precedenza, viene incluso un elemento `hidden` per contenere i valori dell'array `names`:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>
<h2>Questo modulo contiene un array nascosto.</h2>
<input type='hidden' name='names' value='<?php echo($names) ?>'>
<input type='submit' name='submit'>

</form>
```

Funziona tutto molto bene fino a quando non si decide di inviare il modulo e si ottiene l'output della Figura 19.9. Che cosa è accaduto ai dati dell'array? Sfortunatamente non è possibile passare gli array tra i moduli con questa procedura, e pertanto i dati sono andati perduti.

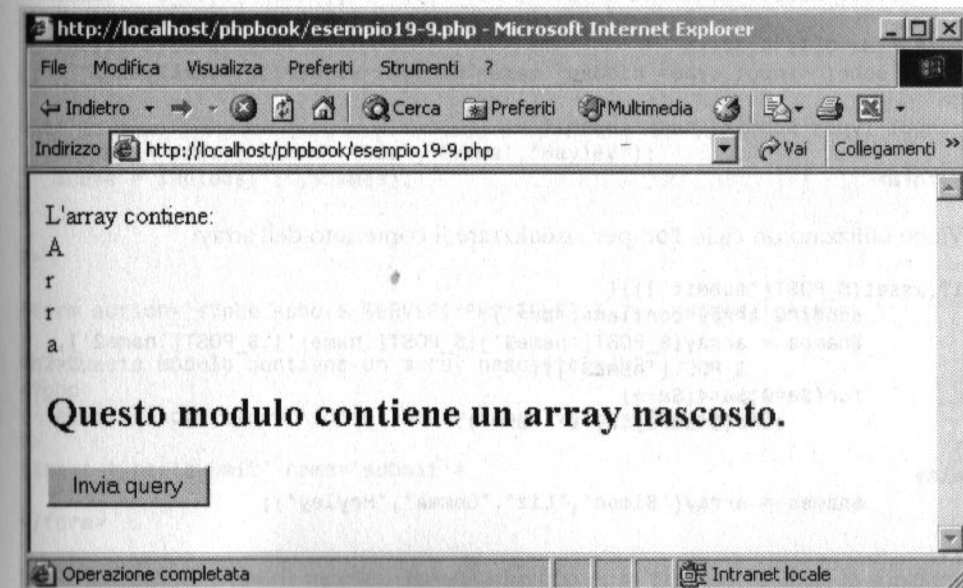


Figura 19.9
Passaggio errato di un array.

Ovviamente esistono metodi che permettono di risolvere il problema.
Come certo ricorderete, un array è una sequenza indicizzata di variabili. Se dividessimo l'array in parti e visualizzassimo ogni elemento dell'array come una variabile

separata, potremmo passare gli array attraverso il modulo. Dopo l'invio del modulo sarà necessario ricombinare i dati, ma il contenuto dell'array sarà mantenuto. Considerate lo script seguente, che è una versione modificata di quello precedente. La prima parte controlla che il modulo sia stato inviato; in caso affermativo, l'array `names` viene creato partendo dalle variabili separate che saranno costruite più avanti:

```
<?php

// Convalida dei moduli e conservazione dei dati - Esempio 19-10
//--------------------------------

if(isset($_POST["submit"])){
  echo("L'array contiene:<br>");
    $names = array($_POST['name0'],$_POST['name1'],$_POST['name2'],
                   $_POST['name3']);
    for($a=0;$a<4;$a++)
      echo($names[$a] . "<br>");
}
else
      $names = array("Simon","Liz","Gemma","Hayley");

?>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo contiene un array nascosto.</h2>
<?php
  for($a=0;$a<4;$a++)
    echo("<input type='hidden' name='name$a' value='", $names[$a]."'>");
?>
<input type='submit' name='submit'>

</form>
```

Viene utilizzato un ciclo `for` per visualizzare il contenuto dell'array:

```
if(isset($_POST["submit"])){
      echo("L'array contiene:<br>");
      $names = array($_POST['name0'],$_POST['name1'],$_POST['name2'],
               $_POST['name3']);
      for($a=0;$a<4;$a++)
          echo($names[$a] . "<br>");
}
else
      $names = array("Simon","Liz","Gemma","Hayley");

?>
```

Come nell'esempio precedente, la parte successiva dello script visualizza il modulo:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo contiene un array nascosto.</h2>
```


La parte seguente dello script utilizza un ciclo `for` per visualizzare i valori dell'array come elementi di modulo nascosto. Osservate che a ciascun elemento è assegnato un nome diverso, partendo da `name0`, `name1` e così via:

```
<?php
  for($a=0;$a<4;$a++)
    echo("<input type='hidden' name='name$a' value='", $names[$a]."'>");
?>
<input type='submit' name='submit'>
</form>

</form>
```

L'esempio precedente funziona, ma esiste un altra procedura che consente di ottenere il medesimo risultato in modo più elegante. Considerate lo script seguente che esegue le stesse operazioni con una tecnica diversa:

```
<?php
// Convalida dei moduli e conservazione dei dati - Esempio 19-11
//--------------------------------

if(isset($_POST["submit"])){
  echo("L'array contiene:<br>");
  $name = $_POST['name'];
  $names = explode("¦",$name);
  for($a=0;$a<4;$a++)
    echo($names[$a] . "<br>");
}
else {
  $names = array("Simon","Liz","Gemma","Hayley");
  $name = implode("¦",$names);
}

?>

<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo contiene un array nascosto.</h2>
<?php
  echo("<input type='hidden' name='name' value='$name'>");
?>
<input type='submit' name='submit'>

</form>
```

Lo script controlla che il modulo sia stato inviato e, in tal caso, utilizza la funzione `explode()` (introdotta nel Capitolo 16) per estrarre un'unica stringa con i nomi separati da caratteri "¦" e inserirla in un array, che sarà poi visualizzato con un ciclo `for`:

```
if(isset($_POST["submit"])){
  echo("L'array contiene:<br>");
  $name = $_POST['name'];
  $names = explode("¦",$name);
  for($a=0;$a<4;$a++)
```

```
    echo($names[$a] . "<br>");
}
```

Se il modulo non è stato inviato, viene creato l'array e viene chiamata la funzione `implode()` per costruire un'unica stringa con il contenuto dell'array, separato con il carattere "¦":

```
else {
   $names = array("Simon","Liz","Gemma","Hayley");
   $name = implode("¦",$names);
}
?>
```

Il modulo viene visualizzato come prima:

```
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

<h2>Questo modulo contiene un array nascosto.</h2>
```

Invece di dover visualizzare diverse variabili nascoste, è sufficiente includere l'unica stringa implosa:

```
<?php
   echo("<input type='hidden' name='name' value='$name'>");
?>
<input type='submit' name='submit'>

</form>
```

La Figura 19.10 mostra l'output dei due script precedenti.

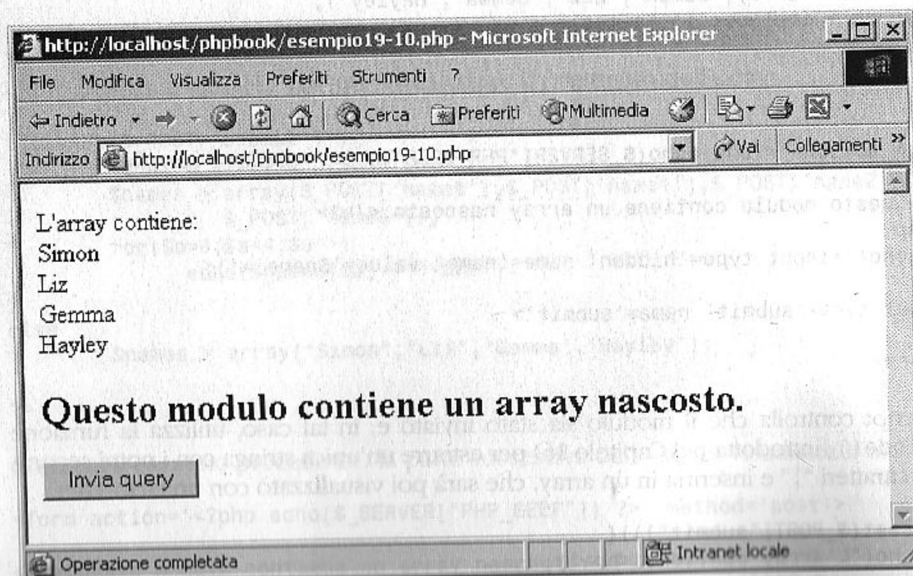


Figura 19.10
Il modo corretto di passare un array.

Riepilogo

Questo capitolo ha mostrato come convalidare i dati dei moduli e ha messo in luce l'importanza di questa operazione per l'interazione con l'utente. Si è visto come sia possibile conservare i dati inseriti correttamente nei moduli nel caso in cui questi debbano essere riproposti agli utenti, eliminando l'onere di inserire di nuovo il testo e selezionare i dati quando non è richiesto. Nel prossimo capitolo verranno analizzati alcuni aspetti della protezione legati all'utilizzo dei moduli, prendendo in considerazione i potenziali rischi che ne possono derivare.

Problemi di sicurezza

Introduzione

Molte applicazioni Web utilizzano metodi di autenticazione o accesso comuni che consentono e impediscono l'accesso a determinati utenti. Tuttavia, il semplice impiego di una password per accedere a un sistema non è l'unico modo per rendere sicuro un sistema di tipo Web. Occorre infatti utilizzare anche metodi di cifratura per memorizzare le password all'interno di database o file di testo; inoltre è possibile verificare le pagine referenti per fare in modo che i dati che vengano passati a una pagina Web provengano effettivamente da un'origine prevista e non da fonte ignota. Questo capitolo illustra come rendere più sicuri i propri sistemi Web online.

Metodi di cifratura

La cifratura aumenta la sicurezza di tutti i sistemi di rete, poiché è in grado di offrire la ragionevole sicurezza che nessuno sarà in grado di sottrarre una password e accedere al sistema in modo non autorizzato. PHP supporta due metodi di cifratura utilizzati comunemente per la cifratura dei dati: MD5 e CRYPT. Tali metodi sono supportati attraverso le due funzioni `md5()` e `crypt()`, e la loro sintassi è la seguente.

```
string md5 (string parola );
string crypt (string parola , string opzione);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
parola	string	Stringa da cifrare.
opzione	string	`CRYPT_STD_DES`. Cifratura DES standard con un sale di due caratteri. `CRYPT_EXT_DES`. Cifratura DES estesa con un sale di nove caratteri. `CRYPT_MD5`. Cifratura MD5 con un sale di dodici caratteri che inizia con **1**. `CRYPT_BLOWFISH`. Cifratura Blowfish con un sale di sedici caratteri che inizia con **2**.
Restituzione di `md5()`/`crypt()`	string	Stringa cifrata.

Esempi di funzioni:

```
$encrypt = md5($text);
$encrypt2 = crypt($text, CRYPT_STD_DES);
```

Lo script dell'esempio seguente illustra l'utilizzo di queste funzioni.

```
<?php

// Problemi di sicurezza - Esempio 20-1
//--------------------

define("TEST","test");

$encrypt = md5(TEST);
$encrypt2 = crypt(TEST, CRYPT_STD_DES);

echo "$encrypt<p>$encrypt2";

?>
```

L'output del testo "test" dopo la cifratura è il seguente.

```
MD5: 098f6bcd4621d373cade4e832627b4f6
Crypt: 1$MDgPcadkr82
```

MD5 è il metodo di cifratura più veloce e più comune. La stringa cifrata è lunga 32 caratteri e non può essere decifrata facilmente. Quando si utilizza CRYPT, la stringa cifrata prodotta è casuale e non può essere sfruttata agevolmente per i metodi di autenticazione Web.

Accesso e autenticazione

L'impiego di moduli sicuri assieme a password cifrate garantisce che gli script offrano la massima sicurezza e consentano l'accesso solo a coloro che dispongono dell'autorizzazione di accesso a questi sistemi. Esistono due tipi di autenticazione: il modulo di accesso semplice e l'autenticazione HTTP. Il secondo metodo utilizza il modulo di autenticazione di Apache ed è più sicuro e molto difficile da violare. Le password devono essere cifrate e memorizzate in forma cifrata; in tal caso, anche se qualcuno leggesse il file delle password, gli sarebbe impossibile utilizzarle in quanto non decifrabili.

Sistema di accesso a modulo semplice che utilizza un file delle password con un singolo utente

Lo script che segue illustra la verifica delle password utilizzando il nome utente "john" e la password cifrata "test":

```
<?php

// Problemi di sicurezza - Esempio 20-2
//--------------------
```



```
if(isset($_POST["submit"])) {

  $userpass = array("john", "098f6bcd4621d373cade4e832627b4f6");

  if(($_POST["username"] == "$userpass[0]") AND
    (md5($_POST["password"]) == $userpass[1]))
  {
    echo "Accesso riuscito!";
  }
}

?>

<html>
  <body bgcolor="#FFFFFF" text="#000000">
    <form name="form1" method="post" action="">

      Nome utente:<input type="text" name="username">
      <br>
      Password:<input type="password" name="password">
      <br>
      <input type="submit" name="submit" value="Accedi">
      <input type="reset" name="Submit2" value="Reimposta">

    </form>
  </body>
</html>
```

Lo script verifica se il modulo è stato inviato:

```
if(isset($_POST["submit"])) {
```

In tal caso l'array delle password viene inizializzato:

```
$userpass = array("john", "098f6bcd4621d373cade4e832627b4f6");
```

Se il nome utente e la password corrispondono, viene visualizzato un messaggio di conferma:

```
  if(($_POST["username"] == "$userpass[0]") AND
    (md5($_POST["password"]) == $userpass[1]))
  {
    echo "Accesso riuscito!";
  }
```

La parte finale dello script crea il modulo per l'utente:

```
<html>
  <body bgcolor="#FFFFFF" text="#000000">
    <form name="form1" method="post" action="">

      Nome utente:<input type="text" name="username">
      <br>
      Password:<input type="password" name="password">
      <br>
      <input type="submit" name="submit" value="Accedi">
      <input type="reset" name="Submit2" value="Reimposta">
```

```
        </form>
    </body>
</html>
```

Autenticazione HTTP sicura con Apache

Quando si utilizza un server Apache, è possibile applicare un'altra forma di autenticazione che sfrutta la funzione `header()` per inviare un'intestazione HTTP al browser.

La sintassi della funzione è la seguente.

```
void header (string httpHeader);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
httpHeader	string	Intestazione HTTP.
Restituzione di `header()`	void	Non restituisce alcunché.

Esempio di funzione:

```
header("WWW-Authenticate: Basic realm=\"Secure Login\"");
```

Lo script seguente illustra un esempio di utilizzo di questa funzione:

```
<?php

// Problemi di sicurezza - Esempio 20-3
//--------------------------------

$userpass = array("john", "098f6bcd4621d373cade4e832627b4f6");

if((isset($_SERVER["PHP_AUTH_USER"])) OR (empty($_SERVER["PHP_AUTH_PW"]))
    OR ($_SERVER["PHP_AUTH_USER"] != $userpass[0])
    OR (md5($_SERVER["PHP_AUTH_PW"]) != $userpass[1])) {

  Header("WWW-Authenticate: Basic realm=\"Secure Login\"");
  Header("HTTP/1.0 401 Unauthorized");
  echo"Accesso fallito. Riprovare";
}
else
  echo "Accesso riuscito";

?>
```

Questo esempio impiega il metodo di autenticazione HTTP per proteggere qualunque accesso al sistema. Viene utilizzata la funzione `header()` per inviare un messaggio "Authentication Required" al browser del client e fare così in modo che venga visualizzata una finestra per l'immissione di nome utente e password. Quando l'utente ha inserito un nome utente e una password, l'URL contenente lo script PHP verrà richiamato con le variabili predefinite `PHP_AUTH_USER` e `PHP_AUTH_PW`, che contengono il nome utente e la password forniti. La finestra a comparsa che viene generata è quella della Figura 20.1.

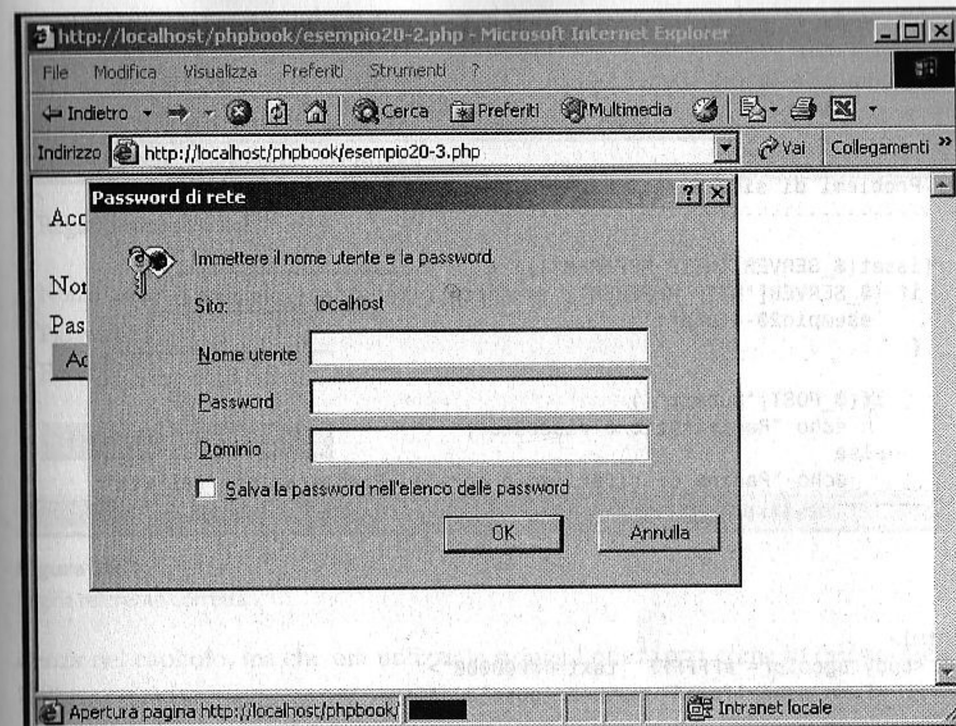


Figura 20.1
Tipo di autorizzazione HTTP standard.

Verifica delle pagine referenti

Quando si sviluppa uno script di registrazione, è necessario adottare la massima sicurezza per proteggere il sistema da registrazioni indesiderate, di massa o automatiche, che potrebbero danneggiare il sistema e il database in cui vengono conservati tutti i nomi utente e le password. La creazione di uno script in grado di generare, per esempio, 100.000 nomi utente casuali per "inondare" l'applicazione con tentativi di accesso è molto semplice.

Immaginate per esempio di avere un modulo Web per la registrazione in linea. Questo modulo è costituito da due caselle di testo (una per il nome utente e l'altra per la password) e da un pulsante per l'invio dei dati. Supponete che la prima casella di testo si chiami "user", che la seconda si chiami "pass" e che il nome del pulsante di invio sia "submit". Quando viene impartita un'istruzione HTTP Post (quando si preme il pulsante Invia) queste tre variabili (`$user`, `$pass`, `$submit`) vengono passate allo script, che registra l'utente.

O meglio, questo è ciò che dovrebbe accedere. Ma che cosa accade se qualcuno (su un altro sito) crea uno script Web che passa le medesime tre variabili al vostro script di registrazione? Che cosa accade se lo script viene eseguito un milione di volte? Potrebbe consentire a qualcuno di introdursi nell'applicazione attraverso nomi utente e password generati automaticamente. La soluzione consiste nel garantire che lo script verifichi il "referente" della pagina, ossia il nome della pagina Web o dello

script che ha passato i dati al vostro script. Se non è il nome che vi aspettate, potete ignorare i dati.
Considerate l'esempio seguente:

```
<?php

// Problemi di sicurezza - Esempio 20-4
//--------------------------------

if(isset($_SERVER["HTTP_REFERER"])) {
   if ($_SERVER["HTTP_REFERER"] == "http://localhost/phpbook/
       esempio20-4.php")
   {

      if($_POST["submit"])
         echo "Registrazione riuscita";
      else
         echo "Pagina di riferimento errata. Registrazione fallita";
   }
}

?>

<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <form name="form1" method="post" action="">

      Nome utente:<input type="text" name="user">
      <br>
      Password:<input type="text" name="pass">
      <br>
      Email: <input type="text" name="email">
      <br>
      <br>
      <input type="submit" name="submit" value="Registra">
      <input type="reset" name="Submit2" value="Reimposta">

      </form>
   </body>
</html>
```

In questo caso la pagina referente è `http://localhost/phpbook/esempio20-4.php`. Nel vostro caso dovrete modificare l'informazione in relazione al nome e al percorso in cui sono memorizzati gli script sul vostro server. Lo script viene diviso in due parti principali: la prima verifica la pagina referente, la seconda visualizza il modulo che l'utente normalmente completerebbe per "registrarsi" con l'applicazione. La Figura 20.2 illustra l'output prodotto dallo script precedente.

Reindirizzamento dell'utente

In molti casi (spesso per ragioni di sicurezza) potreste voler indirizzare automaticamente l'utente a una nuova pagina Web per impedirgli di continuare a utilizzare la pagina corrente. Questo è possibile con la funzione **header()** introdotta in precedenza nel capitolo, ma che ora utilizza la stringa `Location:` come in questo esempio:

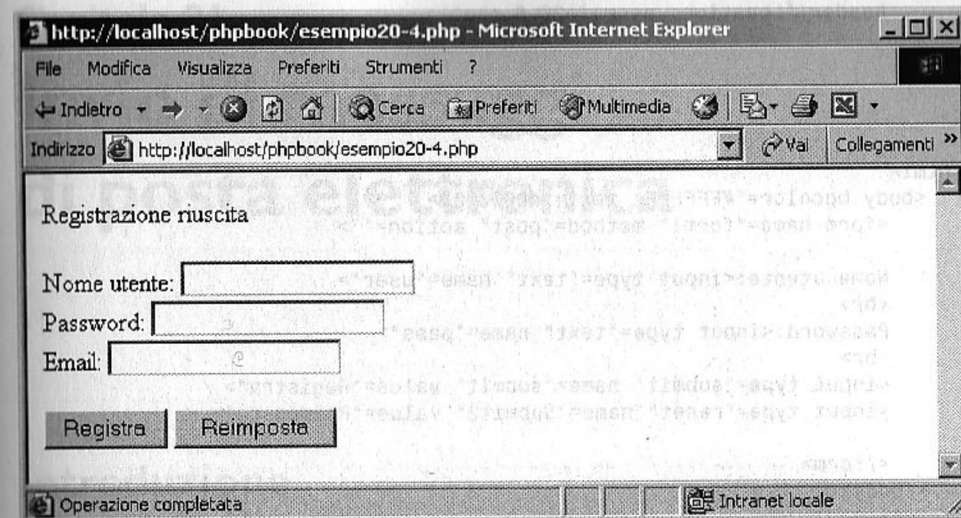


Figura 20.2
Pagina referente corretta.

```
header ("Location: esempio20-5.php");
```

Per vedere un esempio di questo meccanismo, create innanzi tutto lo script seguente:

```
<?php

// Problemi di sicurezza - Esempio 20-5
//--------------------------------

echo "Questa è una pagina diversa";

?>
```

Poi create questo ulteriore script, che è una modifica di esempio20-2.php:

```
<?php

// Problemi di sicurezza - Esempio 20-6
//--------------------------------

if(isset($_POST["submit"])) {

   $userpass = array("john", "098f6bcd4621d373cade4e832627b4f6");

   if(($_POST["username"] == "$userpass[0]") AND
      (md5($_POST["password"]) == $userpass[1]))
   {
      echo "Accesso riuscito!";
   }
   else{
```

```
        header("Location: esempio20-5.php");
    }
}

?>

<html>
    <body bgcolor="#FFFFFF" text="#000000">
        <form name="form1" method="post" action="">

        Nome utente:<input type="text" name="user">
        <br>
        Password:<input type="text" name="pass">
        <br>
        <input type="submit" name="submit" value="Registra">
        <input type="reset" name="Submit2" value="Reimposta">

        </form>
    </body>
</html>
```

L'unica differenza tra quest'ultimo script e quello dell'esempio precedente è che ora, nel caso in cui la password e il nome utente non corrispondano, l'utente viene reindirizzato alla pagina esempio20-5.php.

Riepilogo

Questo capitolo ha introdotto il concetto di protezione delle applicazioni Web mediante sistemi di accesso e registrazione. Sono stati illustrati anche i metodi più comuni per cifrare stringhe e password, analizzando come possono essere implementate per la protezione degli script di autenticazione. Infine è stata introdotta la soluzione della "verifica della pagina referente" per impedire che abbiano luogo registrazioni indesiderate o automatizzate. Nel prossimo capitolo si vedrà come inviare messaggi di posta elettronica agli utenti e come creare messaggi e-mail di solo testo o HTML.

Capitolo 21

Invio di messaggi di posta elettronica

Introduzione

I messaggi di posta elettronica sono diventati uno strumento chiave della comunicazione elettronica. PHP include alcune funzioni per la creazione di messaggi e-mail e per favorire l'interattività con essi. In questo capitolo introdurremo la funzione `mail()`, ampiamente utilizzata negli script PHP per inviare messaggi in formato testo semplice, HTML e MIME a uno o più destinatari contemporaneamente. Oltre a illustrare questa funzione nella sua forma di base, analizzeremo alcune funzionalità più avanzate di `mail()`, tra le quali gli strumenti che consentono di definire intestazioni aggiuntive, modificare il tipo di messaggio inviato, definire gli indirizzi di risposta e altri attributi utili.

Funzione mail()

L'invio di messaggi di posta elettronica con PHP è un'operazione molto semplice. A tal fine si utilizza la funzione `mail()`, con la quale è possibile trasmettere il messaggio da un server di posta di origine a uno specifico account di posta elettronica, sullo stesso server di posta o su uno diverso. Per utilizzare correttamente questa funzione, è necessario avere accesso a un server di posta. Il client di posta da utilizzare è definito nel file php.ini. Di default, la sezione del file php.ini appropriata è la seguente:

```
[mail function]
; For Win32 only.
SMTP = localhost ; for Win32 only

; For Win32 only.

sendmail_from = me@localhost.com ; for Win32 only

; For Unix only. You may supply arguments as well
  (default: "sendmail -t -i").
;sendmail_path =
```

L'ambiente in cui viene eseguito il server PHP, UNIX o Windows, dipende da come viene configurato questo file. Per un utente Windows, dovete configurare l'ubicazione del

server di posta SMTP e l'indirizzo di posta elettronica. Gli utenti UNIX devono inserire l'ubicazione della loro applicazione sendmail, che di default, è /usr/bin/sendmail.
La sintassi della funzione è la seguente.

```
bool mail (string destinatario, string oggetto, string messaggio,
        string intestazioni);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
destinatario	string	Destinatario del messaggio.
oggetto	string	Oggetto del messaggio.
messaggio	string	Il corpo del messaggio.
intestazioni	string	Intestazioni aggiuntive del messaggio.
Restituzione di `mail()`	boolean	TRUE se il messaggio di posta elettronica è stato accettato per la consegna, FALSE in caso contrario.

Esempio di funzione:

```
mail ("qualcuno@esempio.com", "Oggetto", "Riga 1\n Riga 2\n Riga 3");
```

La funzione `mail()` è composta da quattro attributi. Il primo serve per specificare l'account di posta di destinazione al quale inviare il messaggio. Il secondo è il testo che compare nella riga dell'oggetto del messaggio. Il terzo è il testo che costituisce il corpo del messaggio. Infine, il quarto attributo può essere utilizzato per specificare parametri e intestazioni supplementari. Nel prosieguo del libro si vedrà come utilizzare questi attributi, mentre ora imparerete come inviare un semplice messaggio di posta elettronica.

Invio di un messaggio di posta elettronica

È possibile utilizzare questo semplice formato della funzione `mail()` per inviare un breve messaggio di solo testo a un account di posta elettronica. L'account di posta elettronica utilizzato nell'esempio è il seguente: `barbaramega@tiscali.it`. Questo esempio non utilizza il quarto attributo della funzione `mail()`:

```
<?php

//Posta elettronica - Esempio 21-1
//--------------------------------

mail("barbaramega@tiscali.it", "Ciao", "Ciao Barbara, come stai?");

?>
```

Quando eseguite questo script con il browser Web, non vedrete alcun messaggio che confermi l'invio dell'e-mail.
Tuttavia un'analisi della casella *Posta in arrivo* di Barbara rivela che il messaggio spedito dallo script PHP è giunto a destinazione, come mostrato nella Figura 21.1.
Il mittente del messaggio è `barbara@yahoo.it`. PHP utilizza un gateway per gli account di posta elettronica comune a tutti i messaggi, come definito nel file php.ini.

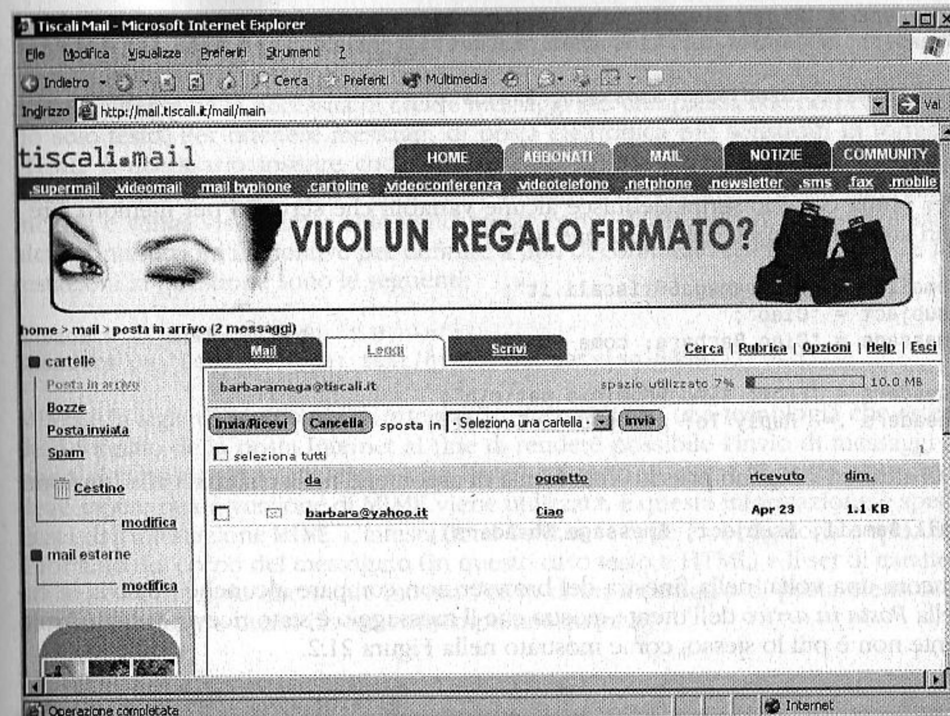


Figura 21.1
Ricezione di un breve messaggio di posta elettronica.

In molti server, il gateway di posta per PHP coincide con l'account di posta elettronica principale del server host (per esempio `root@mailserver.com`).
Per utilizzare un account di posta elettronica personale, dovreste modificare ogni volta il file php.ini oppure (preferibilmente) aggiungere qualche informazione di intestazione in più alla funzione `mail()`.

Invio di messaggi con intestazioni

Il quarto argomento della funzione `mail()` consente di fornire informazioni di intestazioni aggiuntive, relative al messaggio di posta elettronica che desiderate inviare. Se ci sono più intestazioni aggiuntive, occorre separarle con ritorni a capo e caratteri di fine riga (`\r\n`). L'intestazione `From:` specifica il vero indirizzo di posta elettronica del messaggio che vogliamo inviare. L'intestazione `Reply-To:` indica l'indirizzo di risposta che compare quando il destinatario fa clic sul messaggio per inviare una risposta. Lo script che segue illustra l'utilizzo di queste intestazioni.

```
<?php

//Posta elettronica - Esempio 21-2
//------------------

$email = "barbaramega@tiscali.it";
$subject = "Ciao";
$message = "Ciao Barbara, come stai?";
```

```
$headers = "From: simon@dominio.net\r\n";
$headers .= "Reply-To: simon@dominio.net";

mail($email, $subject, $message, $headers);

?>
```

Per prima cosa, lo script definisce alcune variabili che servono per memorizzare i vari attributi richiesti dalla funzione `mail()`:

```
$email =  "barbaramega6@tiscali.it";
$subject = "Ciao";
$message = "Ciao Barbara, come stai?";

$headers = "From: simon@dominio.net\r\n";
$headers .= "Reply-To: simon@dominio.net";
```

Tali attributi vengono passati sotto forma di argomenti nella chiamata alla funzione:

```
mail($email, $subject, $message,$headers);
```

Ancora una volta, nella finestra del browser non compare alcunché, tuttavia la casella *Posta in arrivo* dell'utente mostra che il messaggio è stato ricevuto e che il mittente non è più lo stesso, come mostrato nella Figura 21.2.

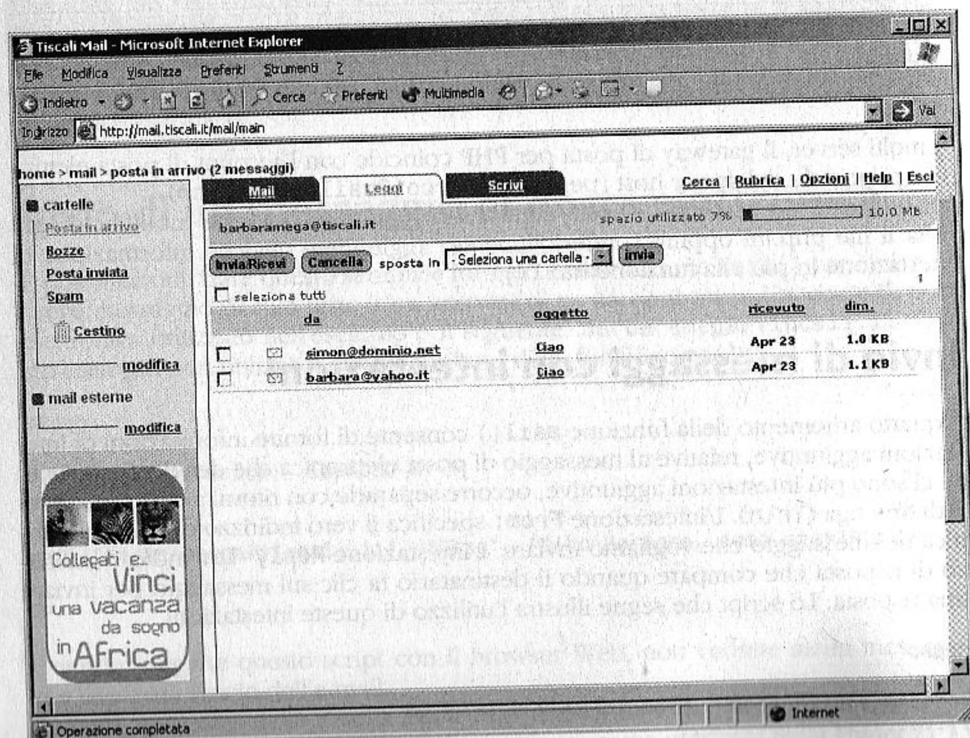


Figura 21.2
Ricezione di un messaggio di posta elettronica con intestazioni supplementari.

Invio di un messaggio HTML complesso

A volte l'utente ha la necessità di creare messaggi più complessi, che non contengano solo testo. Per ottenere messaggi di posta elettronica più sofisticati in formato HTML, è necessario inserire codice HTML nell'attributo `message` della funzione `mail()`. Tuttavia questo non è sufficiente affinché il messaggio di posta elettronica includa e venga visualizzato automaticamente come HTML. Infatti, sono necessarie alcune intestazioni aggiuntive per definire il tipo di contenuto del messaggio. Le intestazioni in questione sono le seguenti:

```
$headers = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";
```

MIME è la sigla di *Multipurpose Internet Mail Extensions*, una tecnologia che estende il formato della posta Internet al fine di rendere possibile l'invio di messaggi di posta elettronica più complessi e non basati sul testo. Il client di posta elettronica deve sapere quale versione di MIME viene utilizzata, e questa informazione è specificata dall'intestazione `MIME`. L'intestazione `Content-type` è obbligatoria e specifica il formato del corpo del messaggio (in questo caso testo e HTML) e il set di caratteri utilizzato. L'esempio che segue mostra l'invio di un messaggio di posta elettronica in stile HTML a tre utenti diversi contemporaneamente:

```
// Invia a
$to  = "barbaramega@tiscali,it, ";
$to .= "kelly@dominio.com, ";
$to .= "mary@dominio.com";

// Oggetto del messaggio
$subject = "Happy Christmas!";

//Messaggio HTML
$message = '
  <html>
    <body bgcolor="#FFFFCC" text="#000000">

      <div align="center">

        <h1>HAPPY NEW YEAR!</h1>
        <br>
        <table width="385" border="0" cellspacing="2" cellpadding="2"
            height="74">

        <tr bgcolor="#CC9900">
          <td bgcolor="#CC0000">Company Meeting:</td>
          <td>20:00</td>
        </tr>

        <tr bgcolor="#CC9900">
          <td bgcolor="#CC0000">New year`s Eve Party:</td>
          <td>22:00</td>
        </tr>

        </table>
```

```
        </div>
    </body>
</html>
';

// Per inviare posta HTML è necessario definire l'intestazione
// Content-type
$headers  = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";

// Intestazione From
$headers .= "From: Company Name <info@company.com>\r\n";

mail($to, $subject, $message, $headers);

?>
```

Innanzitutto, lo script definisce la variabile `$to`, che serve per memorizzare l'elenco dei destinatari dei messaggi, e la variabile `$subject`, che contiene il testo visualizzato nell'oggetto del messaggio:

```
// Invia a
$to  = "barbaramega@tiscali.it, ";
$to .= "kelly@domain.com, ";
$to .= "mary@domain.com";

// Oggetto del messaggio
$subject = "Happy Christmas!";
```

Il blocco successivo di codice definisce il corpo del messaggio HTML, memorizzato nella variabile `$message`:

```
//Messaggio HTML
$message = '
    <html>
        <body bgcolor="#FFFFCC" text="#000000">

            <div align="center">

                <h1>HAPPY NEW YEAR!</h1>
                <br>
                <table width="385" border="0" cellspacing="2" cellpadding="2"
                    height="74">

                <tr bgcolor="#CC9900">
                    <td bgcolor="#CC0000">Company Meeting:</td>
                    <td>20:00</td>
                </tr>

                <tr bgcolor="#CC9900">
                    <td bgcolor="#CC0000">New year`s Eve Party:</td>
                    <td>22:00</td>
                </tr>

                </table>
```



```
            </div>
        </body>
    </html>
';
```

Infine, sono definite le intestazioni del messaggio e viene invocata la funzione `mail()`:

```
// Per inviare posta HTML è necessario definire l'intestazione
// Content-type
$headers  = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";

// Intestazione From
$headers .= "From: Company Name <info@company.com>\r\n";

mail($to, $subject, $message, $headers);

?>
```

Ciascun destinatario riceverà un messaggio di posta elettronica che, una volta visualizzato con il programma di posta, assomiglierà all'esempio mostrato nella Figura 21.3.

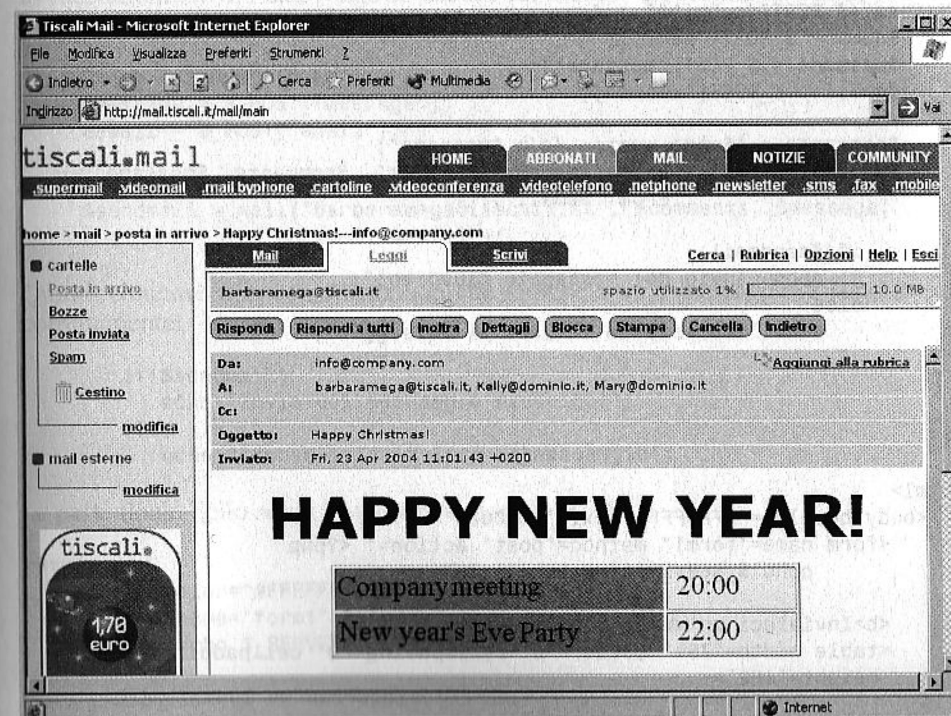


Figura 21.3
Ricezione di un messaggio di posta elettronica HTML complesso.

Creazione di un modulo di posta elettronica

Sempre più spesso, i siti Web mettono a disposizione dell'utente moduli di posta elettronica per inviare messaggi direttamente all'amministratore del sito o a singole persone all'interno di un'organizzazione. In genere questi moduli prendono il nome di "modulo di feedback" o "modulo di posta elettronica". Per creare un tipico modulo di feedback, è sufficiente un semplice modulo HTML con tre o più caselle di testo e un pulsante di invio (in qualche caso è presente anche un pulsante di ripristino per reimpostare tutti i dati contenuti nei campi).
La prima casella di testo contiene il nome dell'utente che invia il messaggio. La seconda contiene il messaggio di posta elettronica dell'utente in modo che il sito possa rispondere, mentre nella terza casella di testo si trova il vero contenuto del messaggio. Ecco un esempio di un modulo simile:

```
<?php

//Posta elettronica - Esempio 21-4
//--------------------

//Se viene premuto il pulsante, invia il messaggio

if(isset($_POST["submit"])) {
    $name = $_POST["name"];
    $message = $_POST["message"];
    $email = $_POST["email"];

    $comments = "$name wrote: /n/n $message";
    $sendmail = mail("barbaramega@tiscali.it", $comments, $message,
                "From: $email");

      if($sendmail)
         echo "Invio del messaggio riuscito!";
      else
         echo "Impossibile inviare il messaggio";

}

?>

<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <form name="form1" method="post" action=" <?php
            echo $_SERVER["PHP_SELF"];?>">

      <b>Inviateci un messaggio:</b><br>
      <table width="75%" border="0" cellspacing="0" cellpadding="0"
       height="182">

         <tr>
            <td>Nome:</td>
            <td><input type="text" name="name"></td>
         </tr>

         <tr>
```



```
            <td>E-mail:</td>
            <td><input type="text" name="email"></td>
         </tr>

         <tr>
            <td>Messaggio:</td>
            <td>
               <textarea name="message" cols="40"rows="4"></textarea>
            </td>
         </tr>

      </table>
      <input type="submit" name="submit" value="Invia messaggio">
      <input type="reset" name="reset" value="Reimposta">

      </form>
   </body>
</html>
```

La prima parte dello script verifica se il modulo è stato inviato:

```
if(isset($_POST["submit"])) {
```

In caso affermativo, i dati vengono estratti dal modulo e la funzione `mail()` invia il messaggio di posta elettronica:

```
    $name = $_POST["name"];
    $message = $_POST["message"];
    $email = $_POST["email"];

    $comments = "$name wrote: /n/n $message";
    $sendmail = mail("barbaramega@tiscali.it", $comments, $message,
                "From: $email");
```

Quindi compare una comunicazione che annuncia se il messaggio è stato inviato correttamente:

```
      if($sendmail)
         echo "Invio del messaggio riuscito!";
      else
         echo "Impossibile inviare il messaggio";
```

La parte finale dello script visualizza il modulo HTML:

```
<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <form name="form1" method="post" action=" <?php
            echo $_SERVER["PHP_SELF"];?>">

      <b>Inviateci un messaggio:</b><br>
      <table width="75%" border="0" cellspacing="0" cellpadding="0"
       height="182">

         <tr>
            <td>Nome:</td>
```

```
            <td><input type="text" name="name"></td>
        </tr>

        <tr>
            <td>E-mail:</td>
            <td><input type="text" name="email"></td>
        </tr>

        <tr>
            <td>Messaggio:</td>
            <td>
                <textarea name="message" cols="40"rows="4"></textarea>
            </td>
        </tr>

    </table>
    <input type="submit" name="submit" value="Invia messaggio">
    <input type="reset" name="reset" value="Reimposta">

    </form>
  </body>
</html>
```

La Figura 21.4 illustra il modulo che l'utente deve utilizzare per inviare un messaggio di posta elettronica.

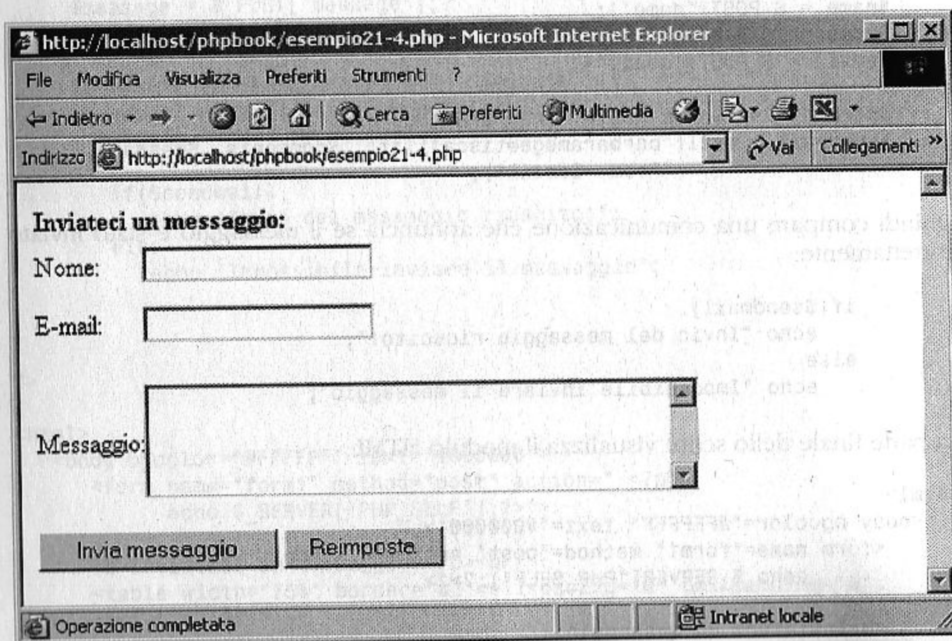


Figura 21.4
Un semplice modulo di posta Web.

Riepilogo

Questo capitolo ha mostrato come utilizzare PHP per inviare un messaggio di posta elettronica a un utente in modo completamente automatico, o con l'interazione da parte dell'utente tramite un modulo di posta Web. Sono stati presentati due stili diversi di messaggi, dando così la possibilità all'utente di inviare sia messaggi semplici sia messaggi più complessi.

Il prossimo capitolo introdurrà il concetto di cookie e mostrerà come i cookie possano essere utilizzati per interagire con l'utente.

Parte V

Conservazione dei dati nel passaggio tra pagine

Capitolo 22

Cookie

Introduzione

Questo capitolo introduce il concetto di cookie, un mezzo per memorizzare dati variabili a livello locale sul computer dell'utente. I cookie possono essere memorizzati sul computer di tutti gli utenti e i dati conservati in ciascuno di questi cookie possono essere diversi. Grazie ai cookie è pertanto possibile memorizzare le preferenze degli utenti in modo da offrire un'esperienza dinamica personalizzata per ciascun utente che accede a un sito Web. Un esempio comune di quanto appena detto sono i siti di e-commerce che consentono a tutti gli utenti di selezionare i diversi prodotti che desiderano acquistare.

Creazione di un cookie

Per la creazione di un cookie è necessario ricorrere alla funzione `setcookie()`. La sintassi della funzione è la seguente.

```
bool setcookie (string nome , string valore , int scadenza , string
percorso, string dominio, int sicuro )
```

Con la funzione `setcookie()` tutti gli argomenti sono facoltativi, tranne il nome. Nel caso in cui sia presente solo l'argomento *nome*, un cookie con quel nome sarà cancellato dal client remoto. Al fine di saltare qualunque argomento, potete anche sostituirlo con una stringa vuota (`""`).
La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nome	string	Nome del file del cookie.
valore	string	Dati da memorizzare nel file del cookie.
scadenza	int	Stringa di data che definisce la durata del cookie.
percorso	string	Sottoinsieme di URL di un dominio per il quale il cookie è valido.

(*Segue*)

Nome	Tipo	Descrizione
dominio	string	Attributi di dominio del cookie realizzati con il nome di dominio Internet dell'host dal quale l'URL verrà raggiunto.
sicuro	boolean	Se impostato a "1" verrà trasmesso solo se il canale di comunicazione con l'host è sicuro.
Restituzione di setcookie()	boolean	TRUE in caso di successo, altrimenti FALSE.

Esempi di funzioni:

```
setcookie ("TestCookie", $value);
setcookie ("TestCookie", $value,time()+3600);
setcookie ("TestCookie", $value,time()+3600, "/nomeutente/",
"dominio.com", 1);
```

Quando utilizzate la funzione per la generazione di cookie, dovete tenere presenti le considerazioni illustrate di seguito.

- I cookie non diventeranno visibili fino al successivo caricamento della pagina per la quale il cookie dev'essere visibile. Per verificare se l'impostazione di un cookie è riuscita, verificatene la presenza al successivo caricamento della pagina prima della sua scadenza. L'ora di scadenza viene impostata con il parametro *scadenza*.
- I cookie devono essere eliminati con gli stessi parametri con i quali sono stati impostati.
- I nomi dei cookie possono essere impostati come nomi di array e saranno disponibili per gli script PHP come array; tuttavia sul sistema dell'utente vengono memorizzati cookie separati. Potreste prendere in considerazione l'utilizzo delle funzioni explode() o serialize() per impostare un cookie con più nomi e valori.

Considerate l'esempio seguente che illustra la funzione setcookie():

```
<?php

// Cookie - Esempio 22-1
//--------------------

$value = "Salve, questo è del testo in un cookie";
setcookie ("TestCookie", $value,time()+3600);

?>
```

L'output di questo script non è molto interessante, in quanto non viene visualizzato alcunché e lo schermo del browser rimane vuoto. Per vedere se l'impostazione del cookie è riuscita, dovete sapere come leggerlo.

Lettura di un cookie

La visualizzazione e l'accesso ai cookie è semplice; è sufficiente trattarli come una variabile predefinita. Nel Capitolo 9 si è accennato al fatto che l'accesso alle variabili dei cookie può avvenire con l'array $_COOKIE, come potete verificare accedendo a esse. Ricordate che i cookie non saranno visibili fino al caricamento della pagina per la quale il cookie deve essere visibile. Per verificare se l'impostazione di un cookie è riuscita, verificatene la presenza al successivo caricamento della pagina prima della sua scadenza. Lo script che segue conferma che è possibile accedere ai cookie creati e quindi visualizzarli.


```
<?php

// Cookie - Esempio 22-2
//--------------------

$value = "Salve, questo è del testo in un cookie";
setcookie ("TestCookie", $value,time()+3600);

?>

<html>
  <body>

   <b>TestCookie contiene:</b> .
   <?php echo($_COOKIE["TestCookie"]); ?>

  </body>
</html>
```

L'output di questo script è quello della Figura 22.1.

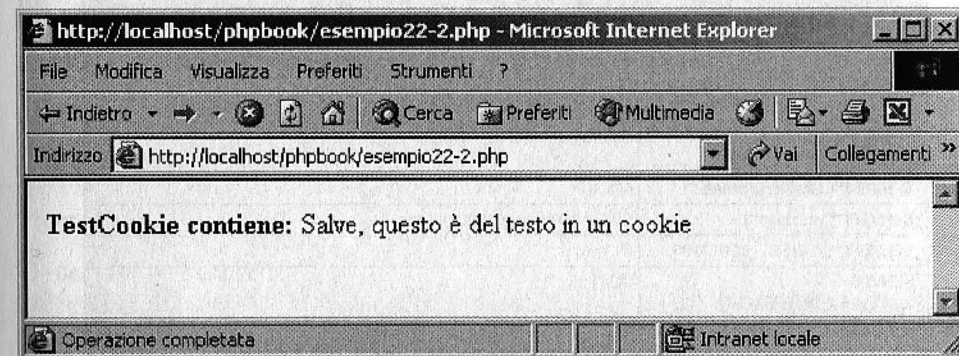


Figura 22.1
Visualizzazione di una variabile cookie.

Un altro modo per verificare se un cookie è stato impostato correttamente e quindi leggerne i dati prevede il ricorso alla funzione phpinfo():

```
<?php

// Cookie - Esempio 22-3
//--------------------

$value = "Salve, questo è del testo in un cookie";
```

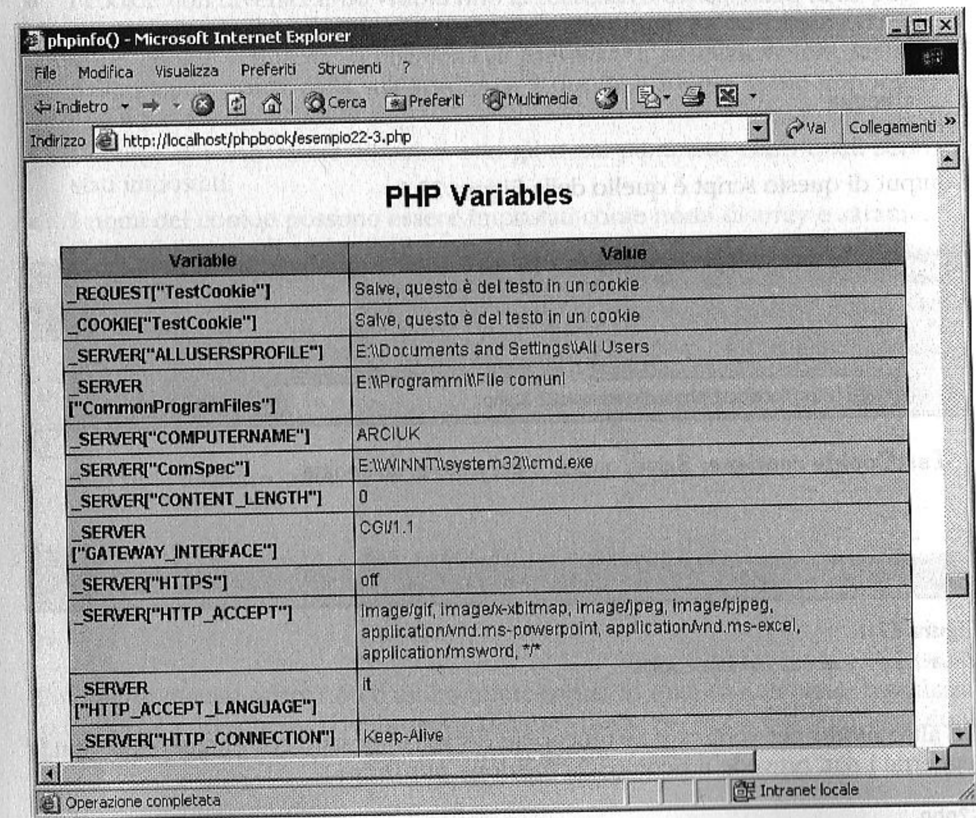
```
setcookie ("TestCookie", $value,time()+3600);
?>

<html>
    <body>

    <?php phpinfo();?>

    </body>
</html>
```

Questo script invoca la funzione `phpinfo()`, che visualizza le informazioni sull'ambiente PHP. Quando le informazioni appaiono sullo schermo, è sufficiente far scorrere la finestra del browser per giungere alla sezione relativa alle variabili PHP. Notate che la seconda e la terza riga nella tabella delle variabili PHP contengono le voci seguenti: `_COOKIE["TestCookie"]`.
La Figura 22.2 illustra quanto appena affermato.



Variable	Value
_REQUEST["TestCookie"]	Salve, questo è del testo in un cookie
_COOKIE["TestCookie"]	Salve, questo è del testo in un cookie
_SERVER["ALLUSERSPROFILE"]	E:\\Documents and Settings\\All Users
_SERVER ["CommonProgramFiles"]	E:\\Programmi\\File comuni
_SERVER["COMPUTERNAME"]	ARCIUK
_SERVER["ComSpec"]	E:\\WINNT\\system32\\cmd.exe
_SERVER["CONTENT_LENGTH"]	0
_SERVER ["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["HTTPS"]	off
_SERVER["HTTP_ACCEPT"]	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
_SERVER ["HTTP_ACCEPT_LANGUAGE"]	it
_SERVER["HTTP_CONNECTION"]	Keep-Alive

Figura 22.2
Visualizzazione dei cookie con phpinfo().

Eliminazione di un cookie

Avete già visto che i cookie possono essere creati con una scadenza incorporata; ma che cosa accade se si desidera eliminare un cookie che non contiene tale scadenza o se lo si vuole cancellare prima di quanto stabilito? Per forzare la scadenza di un cookie, impostate una scadenza negativa utilizzando la stessa sintassi impiegata per la creazione del cookie.

```
setcookie ("TestCookie", "",time()-3600);
setcookie ("TestCookie", "",time()-3600, "/nomeutente/",
"dominio.com", 1);
```

oppure

```
setcookie ("TestCookie");
```

Cookie contenenti array

È possibile memorizzare un array in un cookie, tuttavia è necessario memorizzare separatamente ciascun elemento dell'array nel cookie, come nell'esempio seguente:

```
setcookie ("cookie[0]", "cookietre");
setcookie ("cookie[1]", "cookiedue");
setcookie ("cookie[2]", "cookieuno");
```

Benché gli elementi debbano essere memorizzati separatamente, quando l'array viene letto viene restituito come array al quale è possibile accedere:

```
$list = $_COOKIE["cookie"];
```

Lo script che segue illustra l'impostazione e l'accesso di un array attraverso i cookie:

```
<?php

// Cookie - Esempio 22-4
//......................

$address = $_SERVER['REMOTE_ADDR'];
$browser = $_SERVER['HTTP_USER_AGENT'];
$os = $_SERVER['OS'];

setcookie ("cookie[0]", "$address", time()+200);
setcookie ("cookie[1]", "$browser", time()+200);
setcookie ("cookie[2]", "$os", time()+200);

?>

<html>
    <body>

    <?php

        $list = $_COOKIE['cookie'];
```

```
    echo "<b> Indirizzo IP:</b> $list[0]";
    echo "<br>";
    echo "<b> Browser del client:</b> $list[1]";
    echo "<br>";
    echo "<b> Sistema operativo:</b> $list[2]";

 ?>

  </body>
</html>
```

L'output di questo script è quello della Figura 22.3.

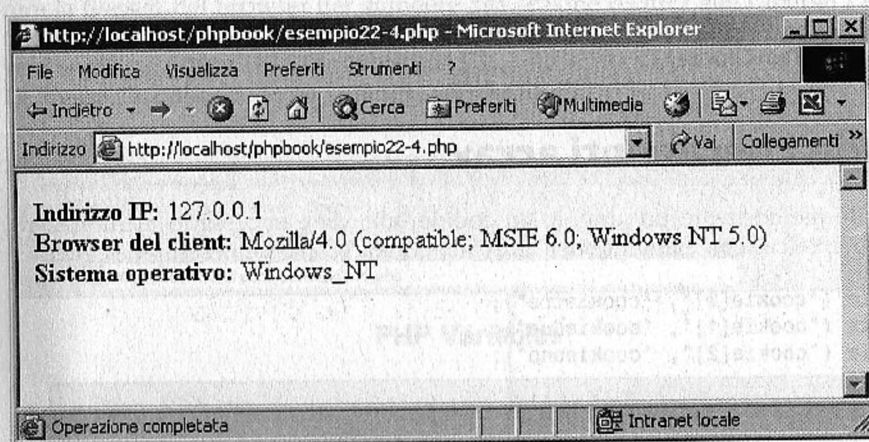


Figura 22.3
Array e cookie.

Cookie a più dati

Un altro metodo per memorizzare più dati in un cookie consiste nel creare una stringa contenente tutte le informazioni preferenziali e passarla come singolo valore al cookie. Si può poi utilizzare la funzione `explode()` per estrarre tutte le informazioni dal cookie. Questa tecnica richiede una struttura più complessa, affinché i dati siano inseriti nel cookie, ma non richiede un array di cookie.
Considerate l'esempio che segue, che è una riscrittura dello script precedente ma utilizza il metodo dei dati multipli.

```
<?php

// Cookie - Esempio 22-5
//--------------------

$address = $_SERVER['REMOTE_ADDR'];
$browser = $_SERVER['HTTP_USER_AGENT'];
$os = $_SERVER['OS'];
$info = "$address::$browser::$os";
```



```
setcookie ("data", "$info", time()+200);

?>

<html>
  <body>

   <?php

     $readcookie = $_COOKIE["data"];
     $list = explode ("::", $readcookie);

     echo "<b> Indirizzo IP:</b> $list[0]";
     echo "<br>";
     echo "<b> Browser del client:</b> $list[1]";
     echo "<br>";
     echo "<b> Sistema operativo:</b> $list[2]";

   ?>

   </body>
</html>
```

Creazione di un sistema per l'accesso

I cookie possono essere utilizzati per creare un sistema di accesso riservato a utenti registrati. Per ottenere il nome utente e la password corretti verrà utilizzato un modulo. In caso di accesso riuscito, il nome utente fornito viene aggiunto da un cookie presente sul computer del client. Il sistema verifica se il cookie esiste: in caso affermativo, viene visualizzato un messaggio "Ciao *nomeutente*", altrimenti compare il messaggio "Nome utente o password non validi":

```
<?php

// Cookie - Esempio 22-6
//--------------------

$username = "simon";
$password = "12345";

if(isset($_POST['login'])){
   if($_POST['user'] == "$username" && $_POST['pass'] == "$password")
   {
      setcookie("member", "$username", time()+600);
   }else{
      echo "Nome utente o password non validi";
   }
}

?>

<html>
```

```
  <head>

  <title>Membri</title>

  </head>
  <body bgcolor="#FFFFFF" text="#000000">

  <?php

    if(isset($_COOKIE['member'])){
       echo "Ciao <b> $_COOKIE[member] </b>";
    }else{
       echo "<b>Accesso non avvenuto </b>";
       echo "<form name='form1' method='post' action=''>";
       echo "Nome utente: <input type='text' name='user'><br>";
       echo "Password: <input type='password' name='pass'><br>";
       echo "<input type='submit' name='login' value='Accedi alla
              pagina riservata'>";
       echo "</form>";
    }

  ?>
  </body>
</html>
```

L'output di questo script è quello della Figura 22.4.

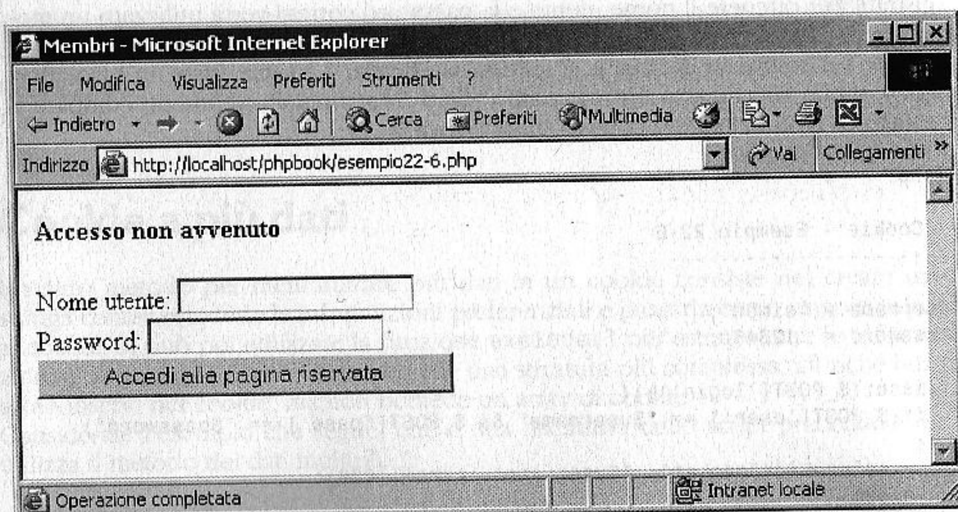


Figura 22.4
Utenti del sistema.

Lo script precedente mostra come consentire agli utenti del sistema di accedere. Lo script che segue può essere impiegato per eseguire l'autenticazione in qualunque pagina del sito riservata a specifici utenti e reindirizzare quelli non autorizzati a una pagina di accesso. Se il cookie esiste, lo script "reimposta" la scadenza del cookie a 10

minuti. Questo script dev'essere collocato su tutte le pagine del sistema dedicato a specifici utenti. Il file dev'essere quindi incluso all'inizio di tutte le pagine.

```
<?php

// Cookie - Esempio 22-7
//--------------------

if(isset($_COOKIE['member'])){
   setcookie("member", "$_COOKIE[member]", time()+600);
}else{
   HEADER("location: esempio22-6.php");
}

?>
```

Questo script utilizza la funzione `header()` per reindirizzare automaticamente l'utente a una nuova pagina Web.
La sintassi della funzione è la seguente.

```
Int header(string indirizzo);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione

Nome	Tipo	Descrizione
indirizzo	string	Indirizzo della pagina Web.
Restituzione di header()	int	Restituisce TRUE se il reindirizzamento va bene, FALSE in caso contrario.

Riepilogo

Questo capitolo ha introdotto il concetto di cookie e ne ha illustrato l'impiego. Si è visto come i cookie possano essere impostati e rimossi, se necessario. Nel prossimo capitolo verrà introdotta una tecnologia simile, quella delle sessioni.

Capitolo 23

Gestione delle sessioni

Introduzione

Nel capitolo precedente avete visto che i cookie possono essere utili nella gestione di informazioni specifiche dell'utente. Tuttavia i cookie possono essere disabilitati e/o cancellati. Ciò di cui avete veramente bisogno è uno strumento con cui creare informazioni specifiche dell'utente (che si conservino nei molteplici accessi Web) e che consenta di eseguire tutte queste operazioni nel modo più automatico possibile. Il supporto alle sessioni in PHP offre questa funzionalità che, come vedrete in seguito, consente di realizzare siti Web molto più interessanti che rendono possibili personalizzazioni più specifiche relative all'utente.

Che cosa sono le sessioni

Le sessioni sono un meccanismo per memorizzare informazioni diverse per ciascun utente che accede al sistema Web. In pratica, le sessioni permettono di memorizzare le variabili e i relativi valori per ogni utente. I valori di queste variabili possono differire da utente a utente, e questo consente di assegnare a utenti diversi potenzialità differenti.

Come funzionano le sessioni

Le sessioni funzionano assegnando a un visitatore un ID univoco, noto come ID di sessione. Tale valore può essere memorizzato direttamente dall'utente in un cookie, oppure viene incorporato come parte dell'URL. Di seguito si riporta un esempio di ID di sessione.

```
sess_f231be97d46fb1ca96c1323e88f4523f
```

Nel server viene creato un file di sessione con lo stesso nome. Questo file serve per memorizzare i valori delle variabili assegnate alla sessione. Il contenuto di un file di sessione potrebbe avere il seguente aspetto:

```
count|i:18;name|s:5:"Simon";
```

In questo esempio, count è una variabile memorizzata nella sessione. La variabile count è di tipo intero e contiene il valore 18; name è invece una variabile di tipo stringa e contiene il valore "Simon".
Quindi, per creare una sessione occorre invocare la funzione session_start().
La sintassi della funzione è la seguente.

```
bool session_start(void)
```

La tabella seguente descrive il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Restituzione di session_start()	boolean	Restituisce TRUE.

Esempio di funzione:

```
session_start();
```

La funzione session_start() controlla se per l'utente in questione è stata creata una sessione e, in caso negativo, ne crea una. Se esiste già una sessione, tutte le variabili e i relativi valori vengono recuperati e resi disponibili. La funzione restituisce sempre TRUE.
Le variabili delle sessioni sono registrate con l'array associativo $_SESSION. Per esempio:

```
$_SESSION['count'] = 0;
```

Questa riga di codice registra, insieme alla sessione, una variabile di nome count. Ciascun utente avrà il proprio file di sessione separato e i propri valori delle variabili. Il paragrafo che segue prende in considerazione un semplice esempio.

Conteggio del numero di accessi

Sfruttando quello che abbiamo imparato finora, possiamo utilizzare le sessioni per accertare quante volte un utente specifico ha tentato di accedere a una pagina. Considerate lo script seguente:

```
<?php

// Sessioni - Esempio 23-1
//-----------------------------

session_start();
if (!isset($_SESSION['count']))
   $_SESSION['count'] = 0;
else
   $_SESSION['count']++;
echo "Ciao, questa pagina è stata visitata da te ". $_SESSION["count"].
     " volte.";

?>
```


Questo script invoca la funzione session_start() sia per creare una nuova sessione sia per accedere a una già esistente. La variabile count è registrata insieme alla sessione e viene impostata a 0 nel caso in cui non sia ancora stata registrata:

```
if (!isset($_SESSION['count']))
   $_SESSION['count'] = 0;
```

In caso contrario, il valore della variabile count viene incrementato e viene utilizzata un'istruzione echo per visualizzarne il contenuto:

```
else
   $_SESSION['count']++;
echo "Ciao, questa pagina è stata visitata da te ". $_SESSION["count"].
     " volte.";
```

Ogni volta che l'utente accede alla pagina, il valore di $count viene incrementato e memorizzato nella sessione. Il risultato è la creazione di un contatore specifico per l'utente, che registra quante volte un determinato utente accede alla pagina. L'output ottenuto da questo script è quello della Figura 23.1.

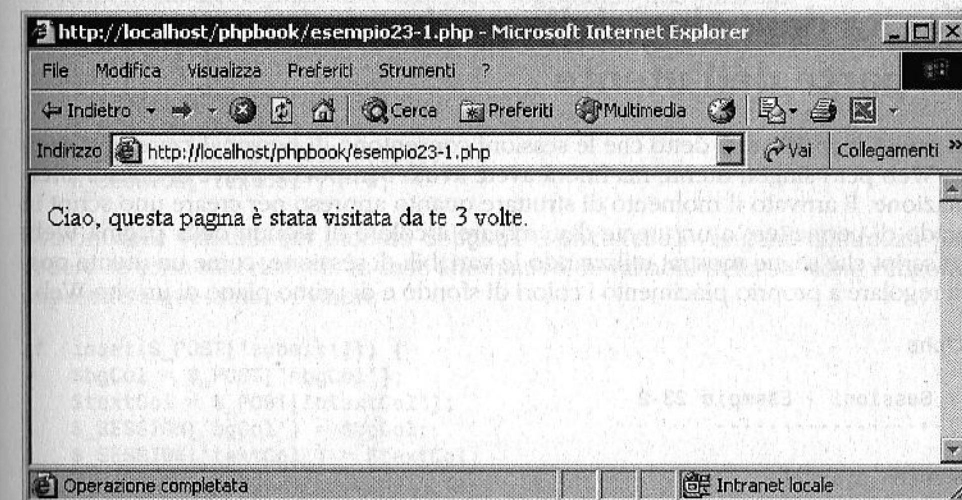


Figura 23.1
Contatore degli accessi alla pagina di un utente.

Ovviamente la Figura 23.1 non dimostra che la sessione sia effettivamente in funzione. Ciò che dovete fare è invitare un altro utente ad accedere alla stessa pagina e vedere se lo script è in grado di tenere traccia dei suoi accessi, senza interferire con i vostri tentativi. Un modo per duplicare questa situazione consiste nell'aprire una nuova finestra del browser e puntarla al medesimo script. Le due finestre del browser dovrebbero gestire gli accessi alla pagina in modo separato per ciascun "utente".

Come ottenere un ID di sessione

Con la funzione `session_id()` è possibile ottenere il valore dell'ID di sessione corrente.
La sintassi della funzione è la seguente.

```
string session_id([string id]);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
id	string	ID di sessione sostitutivo.
Restituzione di session_id()	string	ID di sessione.

Esempio di funzione:

```
$id = session_id();
```

Scelta dei colori della pagina da parte dell'utente

In questo capitolo si è detto che le sessioni consentono di personalizzare una pagina Web per i singoli utenti, ma finora avete avuto ben poche prove di questa affermazione. È arrivato il momento di sfruttare quanto appreso per creare uno script in grado di permettere a un utente di cambiare il colore di sfondo della pagina Web. Lo script che segue mostra, utilizzando le variabili di sessione, come un utente possa regolare a proprio piacimento i colori di sfondo e di primo piano di un sito Web.

```
<?php

// Sessioni - Esempio 23-2
//-------------------

session_start();

if (!isset($_SESSION['bgCol']))
    $_SESSION['bgCol'] = 0;
if (!isset($_SESSION['textCol']))
    $_SESSION['textCol'] = 0;

if (isset($_POST["submit"])) {
    $bgCol = $_POST["nbgCol"];
    $textCol = $_POST["ntextCol"];
    $_SESSION['bgCol'] = $bgCol;
    $_SESSION['textCol'] = $textCol;
    echo("<body bgcolor='$bgCol' text='$textCol'>");
}

?>

<h2>Quali colori preferisci?</h2>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

Colore di sfondo: <select name='nbgCol'>
                  <option>rosso</option>
                  <option>verde</option>
                  <option>blu</option>
                  <option>azzurro</option>
                  <option>giallo</option>
                  </select>
<br>
Colore del testo: <select name='ntextCol'>
                  <option>rosso</option>
                  <option>verde</option>
                  <option>blu</option>
                  <option>azzurro</option>
                  <option>giallo</option>
                  </select>
<br><br>
<input type='submit' name='submit'>
</form>
```

Lo script inizia dichiarando una sessione e registrando due variabili:

```
session_start();

if (!isset($_SESSION['bgCol']))
    $_SESSION['bgCol'] = 0;
if (!isset($_SESSION['textCol']))
    $_SESSION['textCol'] = 0;
```

I valori delle variabili del modulo `$nbgCol` e `$ntextCol` vengono controllati per vedere se sono stati ricevuti. In caso affermativo, le variabili della sessione vengono impostate a questi nuovi valori:

```
if (isset($_POST["submit"])) {
    $bgCol = $_POST["nbgCol"];
    $textCol = $_POST["ntextCol"];
    $_SESSION['bgCol'] = $bgCol;
    $_SESSION['textCol'] = $textCol;
    echo("<body bgcolor='$bgCol' text='$textCol'>");
}
```

La parte restante dello script visualizza l'elemento `body` con gli attributi `bgcolor` e `text` per permettere di modificare il colore dello sfondo e del testo in primo piano. Viene inoltre visualizzato il modulo per dare la possibilità di scegliere il colore di sfondo da un menu di selezione:

```
<h2>Quali colori preferisci?</h2>
<form action='<?php echo($_SERVER["PHP_SELF"]) ?>' method='post'>

Colore di sfondo: <select name='nbgCol'>
                  <option>rosso</option>
                  <option>verde</option>
                  <option>blu</option>
                  <option>azzurro</option>
                  <option>giallo</option>
```

```
                   </select>
<br>
Colore del testo: <select name='ntextCol'>
                   <option>rosso</option>
                   <option>verde</option>
                   <option>blu</option>
                   <option>azzurro</option>
                   <option>giallo</option>
                   </select>
<br><br>
<input type='submit' name='submit'>
</form>
```

L'output ottenuto da questo script è quello della Figura 23.2.

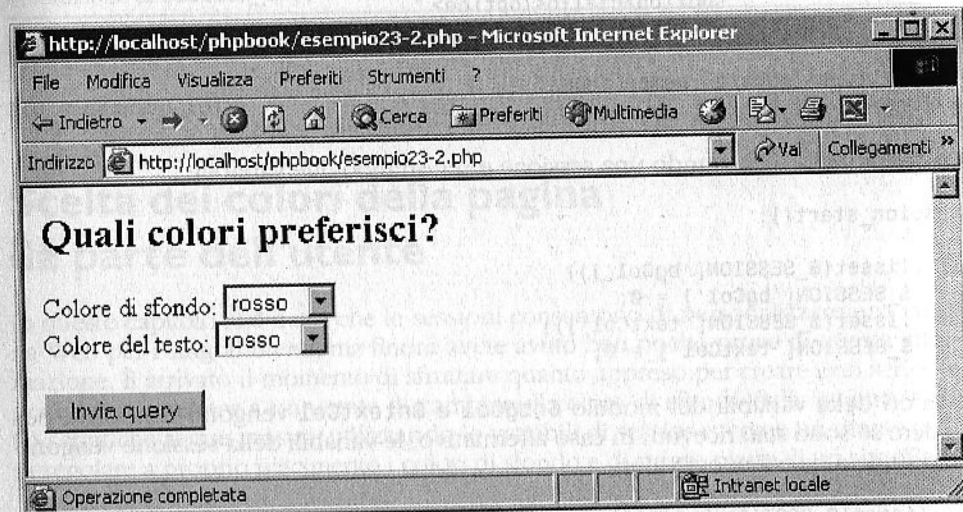


Figura 23.2
Modifica dei colori.

Ancora una volta dovrete aprire una seconda finestra del browser per dimostrare che l'aspetto della sessione della pagina Web sia effettivamente personalizzato per l'utente.

Annullamento della registrazione delle variabili

È possibile annullare la registrazione delle variabili di sessione. Scopo di questa operazione potrebbe essere quello di accertarsi che la sessione creata sia completamente priva di valori di variabili. La funzione `unset()` elimina una variabile registrata insieme alla sessione.
La sintassi della funzione è la seguente.

```
void unset(mixed variabile);
```


La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
variabile	mixed	La variabile per la quale si deve annullare la registrazione.
Restituzione di *unset()*	void	Non restituisce alcunché

Esempio di funzione:

```
unset($_SESSION['bgCol'])
```

Migliore accessibilità alle pagine

Rendere le pagine Web accessibili al maggior numero possibile di utenti è una parte importante della progettazione delle pagine Web. Nell'esempio seguente, vedrete come un utente possa regolare la dimensione del testo di una pagina Web. L'esempio utilizza due pagine Web separate per dimostrare che i valori delle sessioni sono accessibili in più pagine di un unico sito Web. Il primo script, che apre una sessione, dovrebbe essere salvato con il nome "esempio23-3.php":

```
<?php

// Sessioni - Esempio 23-3
//--------------------

session_start();
if (isset($_SESSION['size']))
    $size = $_SESSION['size'];
else
    $size = 3;
echo("<basefont size=$size>");

?>

Benvenuti alla pagina principale. Osservate che il testo viene
visualizzato con una dimensione di font che può essere regolata facendo
clic <a href=esempio23-4.php>qui</a>.
```

Il valore di `size` viene utilizzato per modificare il font di base della pagina:

```
echo("<basefont size=$size>");

?>
```

Gli utenti possono decidere di modificare il font di base facendo clic sul collegamento ipertestuale:

```
Benvenuti alla pagina principale. Osservate che il testo viene
visualizzato con una dimensione di font che può essere regolata facendo
clic <a href=esempio23-4.php>qui</a>.
```

L'output ottenuto da questo script è quello della Figura 23.3.

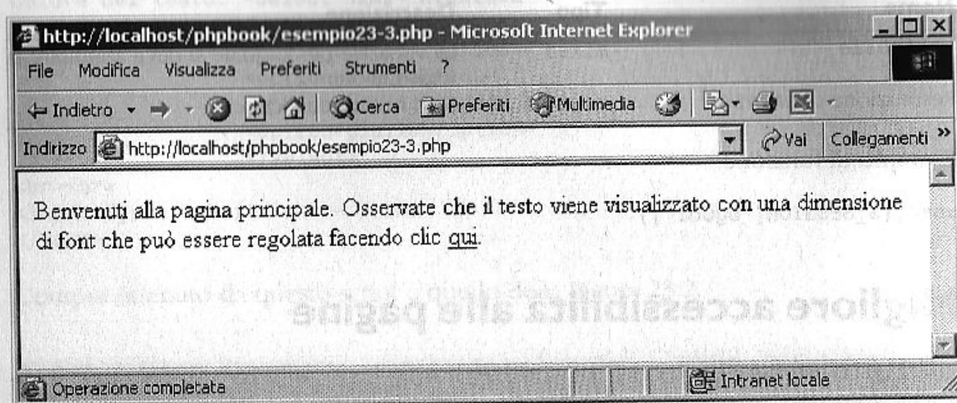


Figura 23.3
Dimensione iniziale del font.

Un clic sul collegamento ipertestuale in questo script invoca lo script seguente:

```
<?php

// Sessioni - Esempio 23-4
//-------------------

session_start();
if (isset($_POST['submit'])) {
    $size = $_POST['size'];
        $_SESSION['size'] = $size;
}
else {
    if (isset($_SESSION['size']))
        $size = $_SESSION['size'];
    else
        $size = 3;
}
echo("<basefont size=$size>");

?>
```

Viene visualizzato un modulo con un menu di selezione che consente all'utente di cambiare la dimensione corrente del font:

```
<form action='<?php echo $_SERVER['PHP_SELF'] ?>' method='post'>

Dimensione del testo: <select name='size'>
                      <option>1</option>
                      <option>2</option>
                      <option>3</option>
                      <option>4</option>
                      <option>5</option>
                      </select>
<br><br>
<input type='submit' name='submit'>

</form>
Torna <a href="esempio23-3.php">Torna</a> alla pagina principale.
```


Lo script deve essere salvato con il nome "esempio23-4.php". Inizia accedendo alla sessione, controlla se la variabile `size` del modulo è stata ricevuta e infine visualizza il font di base con il valore della variabile `$size`:

```
session_start();
if (isset($_POST['submit'])) {
    $size = $_POST['size'];
        $_SESSION['size'] = $size;
}
else {
    if (isset($_SESSION['size']))
        $size = $_SESSION['size'];
    else
        $size = 3;
}
echo("<basefont size=$size>");

?>
```

Compare quindi un modulo con un menu di selezione che consente all'utente di modificare la dimensione corrente del testo:

```
<form action='<?php echo $_SERVER['PHP_SELF'] ?>' method='post'>

Dimensione del testo: <select name='size'>
                      <option>1</option>
                      <option>2</option>
                      <option>3</option>
                      <option>4</option>
                      <option>5</option>
                      </select>
<br><br>
<input type='submit' name='submit'>

</form>
Torna <a href="esempio23-3.php">Torna</a> alla pagina principale.
```

La Figura 23.4 illustra l'output ottenuto da questo script dopo che l'utente ha selezionato 5 come dimensione di font.
Se fate clic sul collegamento ipertestuale *Torna in questa pagina*, tornerete alla pagina esempio23-3.php originale. Il testo di questa pagina avrà una dimensione pari a 5. La dimensione del testo sarà conservata nei passaggi da una pagina all'altra, fino a quando non deciderete di modificarla di nuovo.

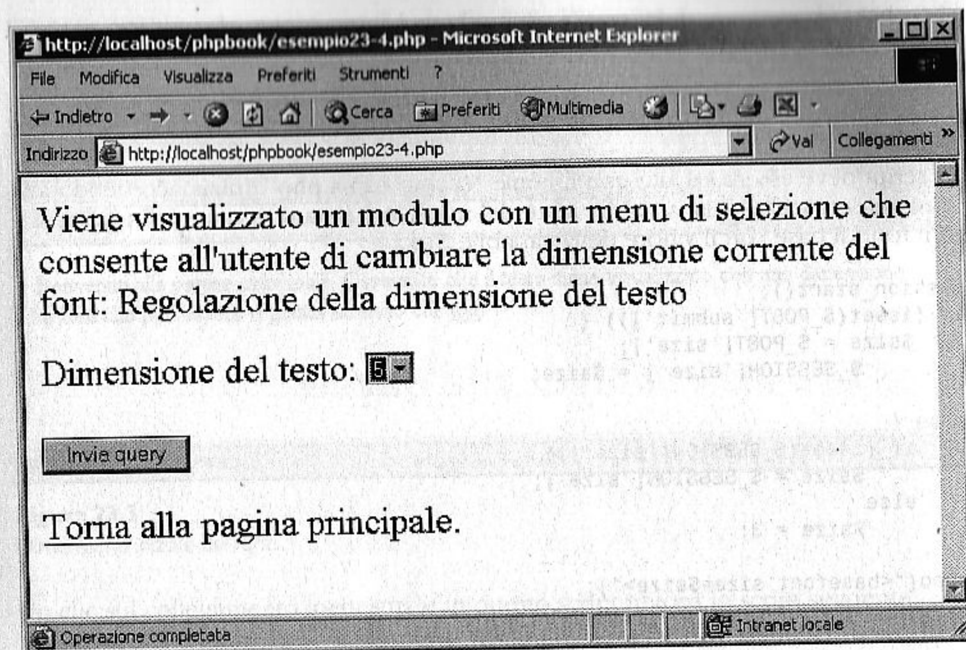


Figura 23.4
È stata selezionata la dimensione di font 5.

Riepilogo

Questo capitolo ha introdotto il concetto di sessione e ha spiegato come creare e memorizzare i valori delle variabili che possono essere modificati per i singoli utenti. Il prossimo capitolo analizzerà il concetto di gestione degli errori e spiegherà come conferire un aspetto più professionale alle applicazioni Web.

Parte VI

Gestione degli errori e buffering

Capitolo 24

Gestione degli errori

Introduzione

Lo sviluppo di codice privo di errori è l'utopia degli ingegneri del software, ma sfortunatamente la ricerca di metodi e strumenti che consentano di raggiungere questo risultato non può dirsi terminata. Nel frattempo, è necessario fare i conti con gli errori dei programmi. Molti linguaggi di programmazione, tuttavia, offrono allo sviluppatore la possibilità di individuare gli errori quando si verificano e quindi agire di conseguenza. Questa azione potrebbe consistere semplicemente nella visualizzazione di un messaggio di errore che avvisa l'utente che è accaduto qualcosa, pur consentendo la prosecuzione dell'esecuzione del programma. In altri casi molto più gravi, invece, il programma potrebbe dover essere interrotto immediatamente.
Questo capitolo presenta le capacità di gestione degli errori di PHP, che consentono non solo di determinare quali errori verranno visualizzati agli utenti, ma anche di scrivere funzioni di gestione degli errori personalizzate. Si vedrà inoltre come innescare i propri messaggi di errore quando occorre e come registrare gli errori per esaminarli in futuro.

Tipi di errore

Normalmente, quando uno script PHP contiene un errore, il parser visualizza un messaggio che indica di che errore si tratta, in quale script si è verificato e in quale linea del codice è stato rilevato. In relazione alla gravità dell'errore, l'esecuzione dello script potrebbe essere interrotta a questo punto, oppure continuare. Mentre i messaggi di errore possono essere tollerati in un ambiente di sviluppo, essi danno un'impressione di scarsa professionalità agli utenti di un ambiente di lavoro.
La cattura degli errori consente di intercettare gli errori quando si verificano, celandone la visualizzazione agli utenti. Se l'errore può essere corretto, ciò in genere avviene (all'insaputa dell'utente) e l'esecuzione dello script può proseguire. Se l'errore è più grave e non può essere corretto, è possibile visualizzare un messaggio nel browser per spiegare che cosa è accaduto e l'esecuzione dello script viene interrotta. PHP supporta tre diversi tipi di errore che differiscono nel livello di gravità. La Tabella 24.1 riporta queste tre tipologie e fornisce esempi di ciò che potrebbe determinarle.

Tabella 24.1 Tipi di errore

Tipo di errore	Descrizione
Notice	Gli avvisi di tipo "Notice" sono generati quando uno script viene eseguito. In genere sono di scarso rilievo (ossia l'interprete PHP è in grado di indovinare ciò che il programmatore intendeva e pertanto l'esecuzione dello script prosegue). Per esempio, un avviso di questo tipo verrebbe generato se utilizzaste una variabile senza prima inizializzarla.
Warning	I messaggi di attenzione "Warning" sono più seri, poiché l'interprete non è in grado di risolvere il problema; tuttavia l'esecuzione dello script può comunque continuare. Per esempio, potreste ottenere un messaggio di questo tipo se cercaste di includere un file che non esiste.
Fatal error	Gli errori irreversibili (Fatal error) sono i più gravi. Quando vengono rilevati, l'interprete non è in grado di continuare l'esecuzione e lo script viene bloccato. Un errore di questo tipo si verificherebbe se cercaste di chiamare una funzione che non è stata dichiarata.

Ciascuno di questi tre tipi di errore può essere generato in diversi momenti da parte di componenti differenti del motore di sviluppo di PHP. Possono essere generati all'avvio del motore PHP, in fase di analisi, in fase di compilazione o durante l'esecuzione dello script. Possono essere prodotti dal nucleo fondamentale del motore di PHP, dalla libreria di funzioni che racchiude questo nucleo o dallo script applicativo. Tutte queste combinazioni producono la definizione di undici diversi tipi di errore, ciascuno dei quali è definito da un valore numerico e da una costante con nome; la Tabella 24.2 ne riporta un elenco.

Tabella 24.2 Valori di errore, costanti e descrizioni

Valore	Costante	Descrizione
1	E_ERROR	Errore irreversibile in esecuzione.
2	E_WARNING	Messaggio "Warning" per un errore non irreversibile in esecuzione.
4	E_PARSE	Errori di analisi in compilazione.
8	E_NOTICE	Avvisi in esecuzione (non gravi come i messaggi "Warning").
16	E_CORE_ERROR	Errori irreversibili all'avvio.
32	E_CORE_WARNING	Messaggio "Warning" per un errore non irreversibile all'avvio.
64	E_COMPILE_ERROR	Errore irreversibile in compilazione.
128	E_COMPILE_WARNING	Messaggio "Warning" per un errore non irreversibile in compilazione.
256	E_USER_ERROR	Messaggio di errore generato dall'utente.
512	E_USER_WARNING	Messaggio "Warning" generato dall'utente.
1024	E_USER_NOTICE	Avviso "Notice" generato dall'utente.
2047	E_ALL	Tutti i precedenti.

Per il momento non preoccupatevi di questi tipi di errore (costanti e valori), poiché li esaminerete più avanti in questo capitolo. Per prima cosa verranno creati alcuni script che genereranno i diversi messaggi di errore per vedere come appaiono.

Esempi di Notice, Warning ed Error

Lo script che segue determinerà la visualizzazione di un avviso "Notice" sulla pagina Web.

```
<?php

// Gestione errori - Esempio 24-1
//-----------------------

$var;
echo("Esecuzione ancora in corso.<br>")

?>
```

Lo script dichiara una variabile `$var` ma non le assegna alcun valore. Lo script genererà il messaggio seguente nel browser:

```
Notice: Undefined variable: var in
c:\inetpub\wwwroot\phpbook\esempio24-1.php on line 6
Esecuzione ancora in corso.
```

Osservate che il messaggio informa del tipo di errore e comunica il nome dello script in cui esso si è verificato e la riga alla quale è stato rilevato. Come potete vedere, l'esecuzione dello script prosegue, in quanto il testo "Esecuzione ancora in corso" viene visualizzato dallo script dopo il rilevamento dell'errore.
Consideriamo ora uno script che genererà un messaggio "Warning" più serio:

```
<?php

// Gestione errori - Esempio 24-2
//-----------------------

include("nonesiste.php");
echo("Esecuzione ancora in corso.<br>");
?>
```

Lo script tenta di includere un file "nonesiste.php" che in questo caso non esiste. Per questo motivo lo script genera il messaggio "Warning" che segue:

```
Warning: main(nonesiste.php): failed to open stream: No such
file or directory in c:\inetpub\wwwroot\phpbook\esempio24-2.php on
line 6

Warning: main(): Failed opening 'nonesiste.php' for inclusion
(include_path='.;c:\php\includes') in
c:\inetpub\wwwroot\phpbook\esempio24-2.php on line 6
Esecuzione ancora in corso.
```

Ancora una volta, osservate che il messaggio di errore segnala il tipo di errore, il nome dello script in cui esso si è verificato e la riga alla quale è stato rilevato. Anche in questo caso il testo "Esecuzione ancora in corso." dimostra che l'esecuzione dello script prosegue.

L'ultimo script di questo paragrafo produce la forma più grave di errore, un messaggio "Fatal error":

```
<?php

// Gestione errori - Esempio 24-3
//----------------------

ball();
echo("Esecuzione ancora in corso.<br>");

?>
```

Lo script provoca un errore irreversibile tentando di chiamare una funzione che non è stata definita. Lo script determina la visualizzazione del messaggio seguente:

```
Fatal error: Call to undefined function: ball() in
c:\inetpub\wwwroot\phpbook\esempio24-3.php on line 6
```

Benché vengano segnalati tipo di errore, script e riga, l'esecuzione termina nel momento in cui l'errore irreversibile viene rilevato e, pertanto, non viene visualizzato alcun testo "Esecuzione ancora in corso".

Regolazione della segnalazione degli errori

Finora avete visto l'output di segnalazione di errore standard generato automaticamente da PHP quando vengono rilevati diversi tipi di errori. PHP consente però di decidere quali dei tre livelli di errore vengono riportati e quali invece vengono semplicemente ignorati. La funzione `error_reporting()` consente di impostare il "livello" corrente di segnalazione degli errori.
La sintassi della funzione è la seguente.

```
int error_reporting(int livello);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
livello	int	Livello di segnalazione degli errori.
Restituzione di `error_reporting()`	int	Restituisce il livello di segnalazione.

Esempio di funzione:

```
error_reporting()
```

La funzione accetta un singolo valore intero o una costante. Per sapere quale valore passare come parametro a questa funzione, è necessario consultare la Tabella 24.2, dove ai diversi tipi di errori sono stati assegnati un valore intero e una costante univoci. La Tabella 24.3 illustra alcuni esempi di combinazione di valori interi e costanti che è possibile passare alla funzione `error_reporting()`. Essa include inoltre una descrizione del livello di segnalazione degli errori fornito dai parametri passati alla funzione.

Tabella 24.3 Esempi di utilizzo della funzione error_reporting()

Esempio	Descrizione
`error_reporting(0)`	Non verrà segnalato alcun errore.
`error_reporting(2047)`	Verranno segnalati tutti gli errori.
`error_reporting(E_ALL)`	Verranno segnalati tutti gli errori (uguale all'esempio precedente).
`error_reporting(3)`	Verranno segnalati solo gli errori `E_ERROR` e `E_WARNING`.
`error_reporting(E_ERROR¦E_WARNING)`	Verranno segnalati solo gli errori `E_ERROR` e `E_WARNING` (uguale all'esempio precedente).

La funzione `error_reporting()` restituisce un valore intero che corrisponde al livello corrente di impostazione della segnalazione di errori. Potete verificare il livello di impostazione corrente chiamando la funzione senza alcun parametro:

```
<?php

// Gestione errori - Esempio 24-4
//----------------------

echo ("Il valore di segnalazione di errori corrente è " .
error_reporting());

?>
```

L'output di questo script è illustrato nella Figura 24.1.

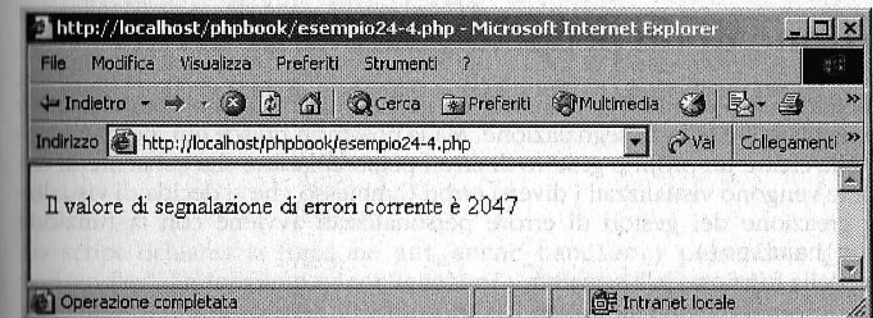


Figura 24.1
Livello di segnalazione degli errori.

Il valore 2047 significa che questo script visualizzerà tutti gli errori, a prescindere dalla loro gravità. Durante lo sviluppo degli script si consiglia di mantenere al livello massimo la segnalazione degli errori, in quanto ciò contribuirà a dar vita a script contenenti il minor numero possibile di errori.

Operatore @

A questo punto vale la pena segnalare che PHP supporta l'operatore @, che può essere utilizzato per disattivare selettivamente i messaggi di errore generati dalle chiamate a funzione. Per esempio:

```
<?php

// Gestione errori - Esempio 24-5
//-----------------------

@ball();
echo("Esecuzione ancora in corso.<br>");

?>
```

L'output di questo script non genererà messaggi di errore di alcun tipo. L'errore, però, viene pur sempre rilevato (la chiamata a una funzione che non è stata definita) e, dato che si tratta di un errore irreversibile, l'esecuzione dello script termina e il messaggio "Esecuzione ancora in corso" non viene visualizzato. Se si rimuove l'operatore @, l'errore irreversibile verrà nuovamente segnalato. Ma perché mai dovreste voler celare i messaggi di errore generati dalla chiamata a una funzione? Spesso le funzioni sono scritte da altri sviluppatori e il loro riutilizzo con il permesso degli autori è un'ottima pratica di programmazione; tuttavia, dato che potreste non poter accedere al nucleo di codice della funzione, nel caso in cui si verifichi un errore, l'unico modo per nasconderlo è ricorrere all'operatore @.

Creazione di un proprio gestore di errori

Finora abbiamo visto che PHP è dotato di un gestore di errori incorporato e che è possibile regolare il livello di segnalazione. Ma la cosa non finisce qui, perché è anche possibile creare un proprio gestore di errori personalizzato, che consente di decidere come vengono visualizzati i diversi errori (ammesso che si decida di visualizzarli). La creazione dei gestori di errore personalizzati avviene con la funzione `set_error_handler()`.
La sintassi della funzione è la seguente.

```
String Set_Error_handler(string gestoreErrori);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
gestoreErrori	string	Il nome della funzione da chiamare quando si verifica un errore.
Restituzione di `set_error_handler()`	string	La stringa `"True"` se la funzione viene eseguita correttamente, `"False"` in caso di problemi.

Alla funzione `set_error_handler()` viene fornito un valore che è il nome di una funzione che verrà chiamata nel caso in cui si verifichi un errore. Questa funzione dev'essere scritta in modo che accetti i parametri riportati nella tabella seguente.

Nome	Tipo	Descrizione
code	string	Il codice dell'errore.
message	string	Una descrizione dell'errore.
filename	string	Lo script in cui l'errore si è verificato.
lineNumber	int	Il numero di riga nello script in cui l'errore si è verificato.
context	array	Un array che punta alla tabella dei simboli attiva nel punto in cui si è verificato l'errore.

La funzione `set_error_handler()`, inoltre, restituisce una stringa, che è la funzione di gestione degli errori attualmente definita (se ne è stata definita una). Lo script seguente illustra un esempio di utilizzo di questa funzione:

```
<?php

// Gestione errori - Esempio 24-6
//---------------------

set_error_handler("errorHandler");

include("nonesiste.php");

function errorHandler($code, $message, $filename, $lineNumber){

   echo("<h2>Sfortunatamente si è verificato un errore</h2>");
   echo("Ecco le informazioni relative all'errore:<br><br>");
   echo("Codice errore: $code<br>");
   echo("Messaggio di errore: $message<br>");
   echo("L'errore si è verificato nel file: $filename<br>");
   echo("L'errore si è verificato alla riga: $lineNumber<br>");

}

?>
```

Lo script richiama la funzione `set_error_handler()` passandole il valore "errorHandler". La funzione `errorHandler()` definita alla fine dello script si limita a visualizzare i valori passati alla funzione dal gestore di errori interno su righe separate. A metà dello script, una funzione `include()` tenta di includere un file che non esiste, generando pertanto un messaggio d tipo "Warning". L'output generato da questo script è illustrato nella Figura 24.2.
Benché la possibilità di creare proprie funzioni di errore personalizzate possa sembrare una scusa per visualizzare l'errore in formati diversi rispetto al gestore di errore standard, esistono altri motivi che possono spingere a crearne una. Per esempio si può decidere di porre termine allo script nel caso in cui vengano rilevati determinati errori.

die ed exit

Esistono due funzioni che consentono di concludere uno script in un dato punto. Spesso queste funzioni vengono impiegate quando viene rilevato un errore di tipo

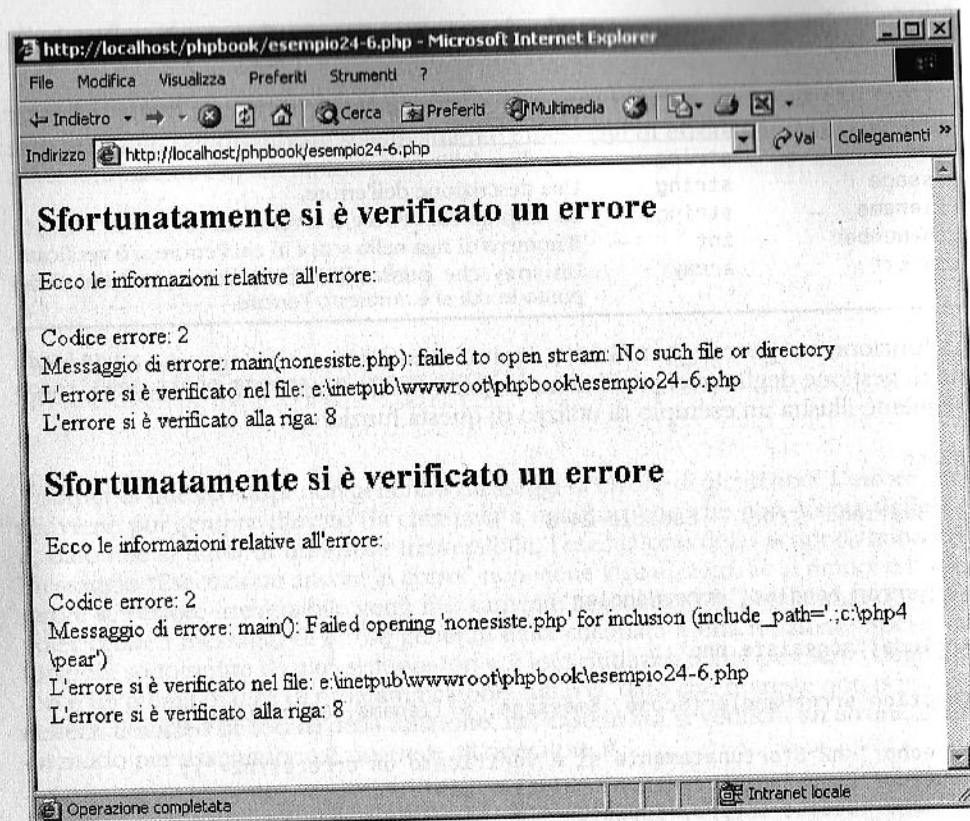


Figura 24.2
Output del gestore di errori personalizzato.

"Warning" o "Notice" e il programmatore non desidera che l'esecuzione dello script prosegua, anche se l'interprete è in grado di farlo.
La sintassi delle funzioni è riportata di seguito.

```
void exit(void);
void die(string messaggio);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
messaggio	`string`	Messaggio da visualizzare prima della terminazione.
Restituzione di `exit()` e `die()`	`void`	Non restituiscono alcunché.

Esempio di funzione:

```
die("Lo script verrà terminato");
```

Entrambe le funzioni concludono uno script, ma la funzione `die()` prima visualizzerà un messaggio nel browser. Lo script seguente illustra l'utilizzo della funzione `die()`. Lo script è uguale al precedente, con la differenza che ora, dopo la visualizzazione del messaggio di errore, lo script viene ultimato.

```
<?php

// Gestione errori - Esempio 24-7
//----------------------

set_error_handler("errorHandler");

include("nonesiste.php");

function errorHandler($code, $message, $filename, $lineNumber){

   echo("<h2>Sfortunatamente si è verificato un errore</h2>");
   echo("Ecco le informazioni relative all'errore:<br><br>");
   echo("Codice errore: $code<br>");
   echo("Messaggio di errore: $message<br>");
   echo("L'errore si è verificato nel file: $filename<br>");
   echo("L'errore si è verificato alla riga: $lineNumber<br>");
   die("Lo script verrà terminato");
}

?>
```

Ovviamente questo script utilizzerà lo stesso stile di output per tutti i tipi di messaggi di errore. Esso inoltre terminerà lo script al rilevamento di qualunque tipo di errore. Tuttavia i gestori di errore personalizzati consentono di avere un maggiore controllo su ciò che avverrà in relazione al tipo di errore rilevato.

Messaggi personalizzati diversi per i vari errori

È pratica comune, tra i programmatori, creare funzioni di gestione degli errori personalizzate che visualizzano diversi tipi e stili di messaggi in relazione al tipo di errore. Il gestore degli errori può anche essere utilizzato per porre termine a uno script, ma solo in caso di rilevamento di determinati tipi di errore. Lo script di esempio che segue modifica la funzione `errorHandler()` per visualizzare diversi messaggi di errore per i tipi di errore `E_NOTICE` e `E_WARNING`.

```
<?php

// Gestione errori - Esempio 24-8
//----------------------

set_error_handler("errorHandler");

$a;
include("nonesiste.php");

function errorHandler($code, $message, $filename, $lineNumber){
```

```
switch($code) {

  case E_NOTICE:
     echo("<br><br>AVVISO: è stato rilevato un errore alla riga
          $lineNumber del file $filename");
     echo("<br>Il messaggio di errore era $message");
     break;

  case E_WARNING:
     echo("<br><br>ATTENZIONE: è stato rilevato un errore alla riga
          $lineNumber del file $filename");
     echo("<br>Il messaggio di errore era $message");
     break;
  }
}

?>
```

Lo script utilizza un'istruzione `switch` per verificare se la variabile `$code` contiene i valori costanti `E_WARNING` e `E_NOTICE`; in relazione a ciò che viene rilevato, viene visualizzato un messaggio di errore diverso. Lo script genera messaggi di errore di tipo "Notice" e "Warning" e l'output generato è quello della Figura 24.3.

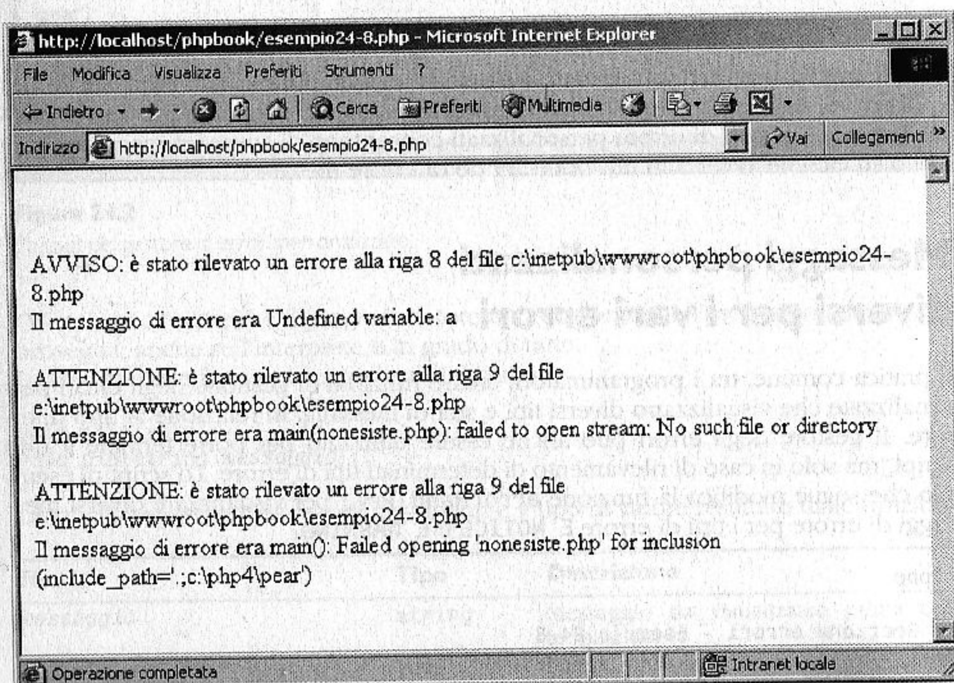


Figura 24.3
Messaggi di errore personalizzati.

Cattura dei tipi di errore E_PARSE ed E_ERROR

Nell'esempio precedente avete visto che la funzione di gestione degli errori personalizzata poteva catturare gli errori `E_NOTICE` ed `E_WARNING`. Non si è provato a catturare anche un messaggio di errore `E_ERROR`, operazione che verrà trattata in questo paragrafo. Lo script che segue è una modifica di quello precedente, ma ora la funzione di gestione degli errori è stata adattata in modo da visualizzare un messaggio di errore quando si imbatte in un errore `E_ERROR`.

```
<?php

// Gestione errori - Esempio 24-9
//------------------------

set_error_handler("errorHandler");

$a;
include("nonesiste.php");
funt();

function errorHandler($code, $message, $filename, $lineNumber){

  switch($code) {

    case E_NOTICE:
       echo("<br><br>AVVISO: è stato rilevato un errore alla riga
            $lineNumber del file $filename");
       echo("<br>Il messaggio di errore era $message");
       break;

    case E_WARNING:
       echo("<br><br>ATTENZIONE: è stato rilevato un errore alla riga
             $lineNumber del file $filename");
       echo("<br>Il messaggio di errore era $message");
       break;

    case E_ERROR:
       die("<br><br>ERRORE IRREVERSIBILE: alla riga $lineNumber del file
            $filename");

  }
}

?>
```

L'output di questo script è quello della Figura 24.4.
Qui è importante osservare che il messaggio di errore irreversibile è stato rilevato, ma non viene visualizzato il messaggio di errore definito nella funzione di gestione degli errori. Il motivo è che gli errori di tipo `E_ERROR` e `E_PARSE` vengono sempre gestiti dal gestore di errori incorporato in php, non da quello personalizzato. Questo vale anche se avete scritto una funzione per la gestione di questo tipo di errori. In altre parole, la modifica apportata alla funzione di gestione degli errori nello script precedente è una perdita di tempo.

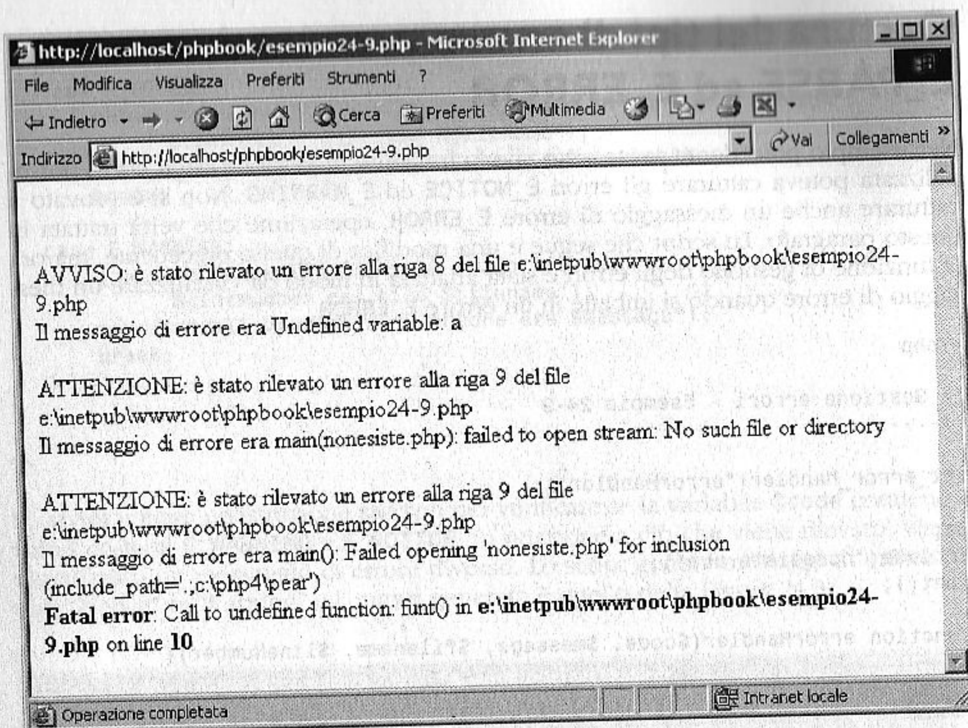


Figura 24.4
Messaggio di errore E_ERROR.

Innesco degli errori

Oltre a creare una propria funzione per la gestione degli errori, PHP consente anche di innescare errori nello script, se lo si desidera. A tal fine è possibile scegliere tra due funzioni, `user_error()` e `trigger_error()`, che sono semplicemente alias una dell'altra.
La sintassi delle funzioni è riportata di seguito.

```
void trigger_error(string messaggio, [int Livello_errore]);
void user_error(string messaggio, [int Livello_errore]);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
messaggio	`string`	Messaggio da visualizzare quando l'errore viene innescato.
Livello_errore	`int`	Valore di errore.
Restituzione di `trigger_error()` e `user_error()`	`void`	Non restituiscono alcunché.

Esempio di funzione:

```
trigger_error("Si è verificato un errore di tipo Notice.<br>");
```

Entrambe le funzioni consentono di sollevare uno qualunque degli errori `E_USER`, ossia `E_USER_NOTICE`, `E_USER_WARNING` e `E_USER_ERROR`; potete trovare i valori interi corrispondenti nella Tabella 24.2. Lo script che segue illustra l'innesco di ognuno di questi diversi tipi di errore.

```
<?php

// Gestione errori - Esempio 24-10
//----------------------

trigger_error("Si è verificato un errore di tipo Notice.<br>");
trigger_error("<br>Si è verificato un errore di tipo Warning.<br>",
              E_USER_WARNING);
trigger_error("<br>Si è verificato un Fatal error.<br>", E_USER_ERROR);

echo("Fine del programma.");

?>
```

Lo script illustra che è possibile creare errori `E_USER_NOTICE` in due modi: si potrebbe fornire alla funzione `trigger_error()` un messaggio di errore insieme a una costante `E_USER_NOTICE`. L'alternativa, illustrata nello script, consiste nel fornire alla funzione `trigger_error()` soltanto un messaggio di errore, che automaticamente porterà alla creazione di un messaggio di livello `E_USER_NOTICE`. Per creare messaggi `E_USER_WARNING` ed `E_USER_ERROR` è necessario specificare il livello di errore come secondo parametro della funzione `trigger_error()`. L'output di questo script è quello della Figura 24.5.

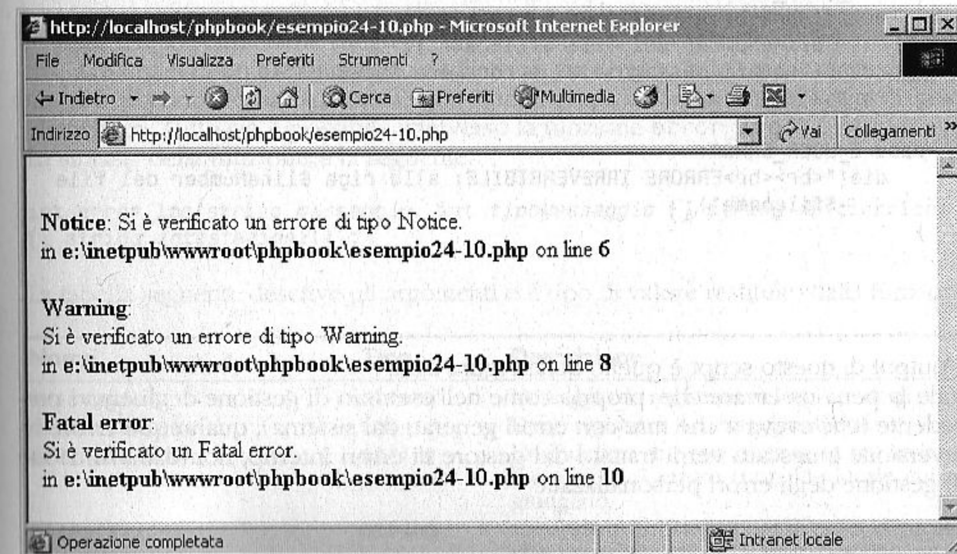


Figura 24.5
Innesco di errori E_USER.

Nell'esempio precedente è il gestore di errori incorporato che gestisce gli errori innescati. Come è facile prevedere, è possibile combinare l'innesco degli errori tramite la

funzione `error_trigger()` con la gestione degli errori personalizzata mediante la funzione `errorHandler()`. Lo script che segue illustra una funzione di gestione degli errori in grado di elaborare errori di tipo `E_USER_NOTICE` ed `E_USER_WARNING`.

```
<?php

// Gestione errori - Esempio 24-11
//---------------------

set_error_handler("errorHandler");

trigger_error("Si è verificato un errore di tipo Notice.");
trigger_error("Si è verificato un errore di tipo Warning.",
              E_USER_WARNING);
trigger_error("Si è verificato un Fatal error.", E_USER_ERROR);

echo("Fine del programma.");

function errorHandler($code, $message, $filename, $lineNumber){

   switch($code) {

   case E_USER_NOTICE:
      echo("<br><br>AVVISO: è stato rilevato un errore alla riga
         $lineNumber del file $filename");
      echo("<br>Il messaggio di errore era $message");
      break;

   case E_USER_WARNING:
      echo("<br><br>ATTENZIONE: è stato rilevato un errore alla riga
         $lineNumber del file $filename");
       echo("<br>Il messaggio di errore era $message");
       break;

   case E_USER_ERROR:
      die("<br><br>ERRORE IRREVERSIBILE: alla riga $lineNumber del file
        $filename");
   }
}

?>
```

L'output di questo script è quello della Figura 24.6.
Vale la pena osservare che, proprio come nell'esempio di gestione degli errori precedente (che aveva a che fare con errori generati dal sistema), qualunque errore irreversibile innescato verrà trattato dal gestore di errori interno, non dalla funzione di gestione degli errori personalizzata.

Registrazione degli errori

Invece di limitarvi a visualizzare il messaggio, potreste voler registrare da qualche parte che l'errore si è verificato. La registrazione degli errori in quelli che vengono definiti "file di registro degli errori" è una pratica molto diffusa. Il registro degli errori fornisce una registrazione permanente degli errori che si sono verificati e di tutte

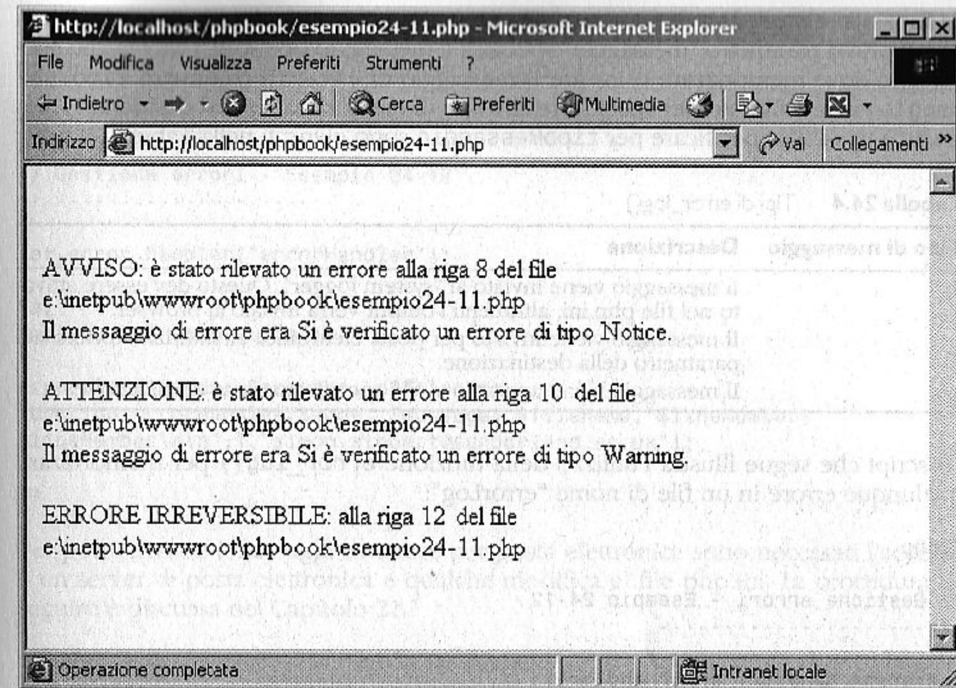


Figura 24.6
Innesco di errori mediante la gestione degli errori.

le volte che lo script viene eseguito. PHP consente di registrare le informazioni relative agli errori in un file di registro degli errori predefinito o in un file di vostra scelta; può persino inviare un messaggio contenente l'errore a un indirizzo di posta elettronica. Tutto ciò è possibile attraverso la funzione `error_log()`.
La sintassi della funzione è la seguente.

```
int error_log(string messaggio, int tipoMessaggio [, string destinazione
[, string intestazioni]]);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
messaggio	string	Messaggio da registrare nel registro degli errori.
tipoMessaggio	int	Dove registrare il messaggio di errore.
destinazione	string	Un indirizzo di posta elettronica cui inviare il messaggio di errore o un file locale cui aggiungerlo.
intestazioni	string	Facoltativo. Le intestazioni di posta elettronica che è possibile utilizzare sono descritte nel Capitolo 21 in relazione alla funzione `mail()`.
Restituzione di `error_log()`	int	1 se la scrittura riesce, `0` in caso di errore.

Esempio di funzione:

```
error_log("$code:$message:\n$filename,$lineNumber\n\n",3,"errorLog");
```

La funzione `error_log()` può assumere un numero variabile di parametri, da due in poi. Il primo parametro "messaggio" specifica il messaggio di errore. I tre restanti specificano dove memorizzare o inviare il messaggio di errore. Il parametro `tipoMessaggio` viene utilizzato per specificare che tipo di memorizzazione si desidera. I valori che è possibile specificare per `tipoMessaggio` sono elencati nella Tabella 24.4.

Tabella 24.4 Tipi di error_log()

Tipo di messaggio	Descrizione
0	Il messaggio viene inviato al "system logger". Questo dev'essere attivato nel file php.ini, altrimenti l'output verrà inviato al browser.
1	Il messaggio viene inviato per posta elettronica all'indirizzo fornito nel parametro della destinazione.
3	Il messaggio viene aggiunto al file specificato come destinazione.

Lo script che segue illustra l'utilizzo della funzione `error_log()` per memorizzare qualunque errore in un file di nome "errorLog":

```
<?php

// Gestione errori - Esempio 24-12
//----------------------

set_error_handler("errorHandler");

$a;
include("nonesiste.php");

function errorHandler($code, $message, $filename, $lineNumber){

   error_log("$code: $message:\n$filename, $lineNumber\n\n",3,"errorLog");

}

?>
```

Il file errorLog verrà salvato nella stessa directory predefinita dei file in cui si trovano gli script o, più esattamente, nella directory in cui lo script viene eseguito. La Figura 24.7 illustra il contenuto del file errorLog dopo la prima esecuzione dello script.

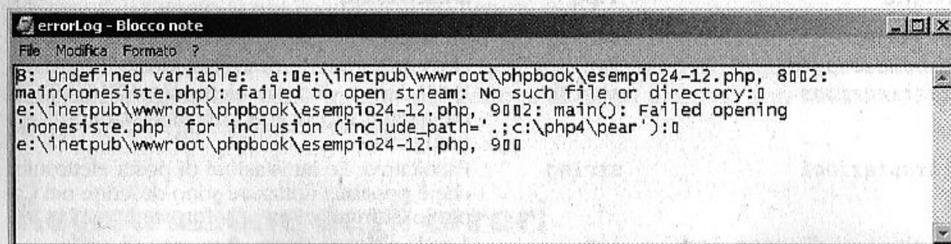


Figura 24.7
Il file errorLog.

Se avete accesso a un server di posta elettronica, potete inviare i messaggi di errore tramite e-mail, come illustrato nello script seguente:

```
<?php

// Gestione errori - Esempio 24-13
//----------------------

set_error_handler("errorHandler");

$a;
include("nonesiste.php");

error_log("$code: $message:\n$filename,
function errorHandler($code, $message, $filename, $lineNumber)
$lineNumber\n\n",1,"simon.stobart@sunderland.ac.uk");

?>
```

Per poter inviare i messaggi di errore per posta elettronica sono necessari l'accesso a un server di posta elettronica e qualche modifica al file php.ini. La procedura da seguire è discussa nel Capitolo 21.

Riepilogo

In questo capitolo è stato illustrato il concetto di gestione degli errori. Si è visto come il gestore di errori nativo di PHP possa catturare gli errori e come il programma possa creare funzioni di gestione degli errori personalizzate. Si è anche visto come sia possibile creare un registro per memorizzare in modo permanente gli errori che si verificano. Nel prossimo capitolo verrà esaminato il concetto di buffering dell'output, mettendo in luce come questa caratteristica possa essere di aiuto nella gestione degli errori.

Capitolo 25

Buffering dell'output

Introduzione

In genere gli script PHP inviano il proprio output direttamente alla periferica di output standard, che di solito è un browser Web. Tuttavia questa non è sempre la soluzione migliore. Supponete, per esempio, che lo script rilevi la presenza di un errore dopo aver visualizzato parte di una pagina Web; anche se errori simili possono essere corretti, come dimostrato nel capitolo precedente, il messaggio di errore potrebbe apparire poco professionale, poiché una parte dell'output è già stata inviata al browser. Questo capitolo introdurrà il concetto di buffering, che permette di memorizzare tutto l'output generato dallo script e manipolarlo a proprio piacimento.

Tutto bene finché non si scopre un errore

Lo script riportato di seguito visualizza un'intestazione e il testo di una pagina Web. È stata definita una funzione per la gestione degli errori, che cattura qualunque errore.

```
<?php

// Buffering dell'output - Esempio 25-1
//----------------------

set_error_handler("catchError");
echo("<h1>Benvenuti nel mondo del buffering.</h1>");
echo("Il buffering è utile per evitare la visualizzazione parziale di una
      pagina Web nel caso in cui si verifichi un errore.");

function catchError() {
   echo("<br>Sfortunatamente si è verificato un errore!.");
}
?>
```

In questo script non ci sono errori quindi, per mostrare cosa accade quando se ne scopre uno, è necessario introdurlo.

```
<?php

// Buffering dell'output - Esempio 25-2
//----------------------

set_error_handler("catchError");
echo("<h1>Benvenuti nel mondo del buffering.</h1>");
echo("Il buffering è utile per evitare la visualizzazione parziale di una
     pagina Web nel caso in cui si verifichi un errore.");
echo($doesnotexist);

function catchError() {
   echo("<br>Sfortunatamente si è verificato un errore!");
}

?>
```

Nello script, un'istruzione `echo` cerca di visualizzare una variabile inesistente, generando così un errore. Quando compare l'errore, la funzione `catchError()` viene invocata e compare il messaggio d'errore appropriato. L'output ottenuto da questo script è quello della Figura 25.1.

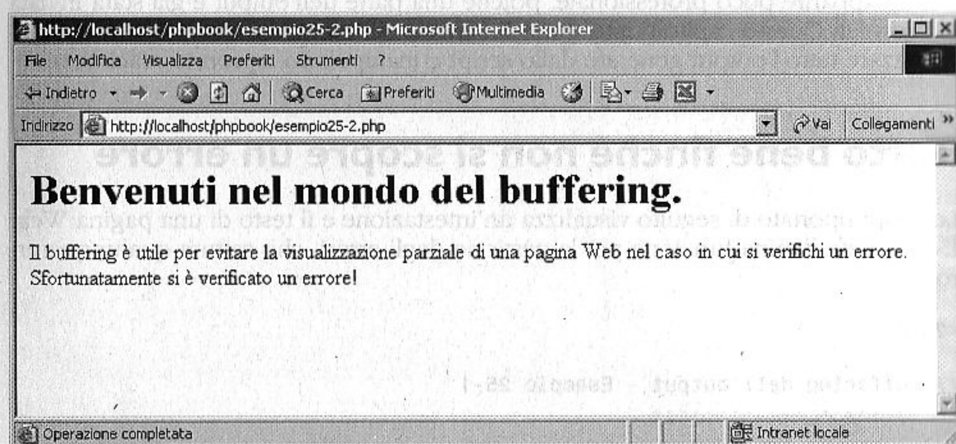


Figura 25.1
Messaggio d'errore visualizzato con altri testi.

La Figura 25.1 mostra che è stato catturato l'errore e visualizzato il messaggio relativo.
Compare anche l'output inviato al browser prima del rilevamento dell'errore. Ovviamente questa è una situazione indesiderabile. Per fortuna PHP permette di controllare l'output attraverso il buffering e, in tal modo, consente di scegliere quali informazioni inviare all'utente.

Buffering dell'output

La funzione `ob_start()` serve per attivare il buffering dell'output. Quando il buffer è stato attivato, lo script non invia alcun output al browser, ma lo conserva in un buffer interno.
La sintassi della funzione è la seguente.

```
Void ob_start([string funz])
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
funz	string	Invocazione della funzione facoltativa.
Restituzione di `ob_start()`	void	Non restituisce alcunché.

La funzione `ob_start()` può accettare un nome di funzione facoltativo per invocare l'avvio del buffering. Più avanti in questo capitolo sarà spiegato come utilizzare questo meccanismo.
L'output viene memorizzato nel buffer e sullo schermo non compare alcunché fino all'invocazione della funzione `ob_end_flush()`, che invierà il contenuto del buffer al browser.
La sintassi della funzione è la seguente.

```
void ob_end_flush(void);
```

La funzione non richiede parametri e non restituisce alcunché. Dopo che il contenuto è stato inviato al browser, il buffering dell'output viene disattivato. Infine, prima di analizzare un esempio dell'utilizzo di queste funzioni, occorre introdurre un'altra funzione: `ob_clean()`.
La sintassi della funzione è la seguente.

```
Void ob_clean(void);
```

Questa funzione cancella tutti i dati contenuti nel buffer di output. Anche questa funzione non richiede parametri e non visualizza alcunché. Considerate un esempio in cui si utilizza il buffering. Nello script seguente, si attiva un buffer e si registra una funzione per la gestione degli errori:

```
<?php

// Buffering dell'output - Esempio 25-3
//----------------------

ob_start();
set_error_handler("catchError");
echo("<h1>Benvenuti nel mondo del buffering.</h1>");
echo("Il buffering è utile per evitare la visualizzazione parziale di una
pagina Web nel caso in cui si verifichi un errore.");

echo($doesnotexist);
ob_end_flush();

function catchError() {
```

```
    ob_clean();
    echo("<br>Sfortunatamente si è verificato un errore!");
}
?>
```

L'output dello script è memorizzato nel buffer. La funzione `ob_end_flush()` del buffer, tuttavia, non viene mai raggiunta, poiché nella riga precedente c'è un errore che comporta l'attivazione della funzione `catchError()`, che cancella il contenuto del buffer e visualizza un messaggio d'errore. L'output prodotto da questo script è quello della Figura 25.2.

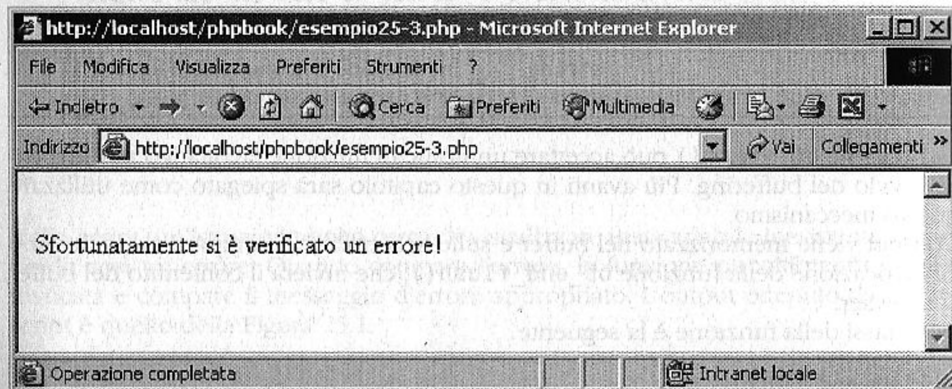


Figura 25.2
Solo un messaggio d'errore.

La Figura 25.2 mostra che, anche se lo script ha prodotto output, è stato possibile intercettarlo, cancellarlo e visualizzarne dell'altro nel browser.

Accesso al contenuto del buffer

Potreste comunque voler accedere e visualizzare il contenuto del buffer, operazione possibilile tramite la funzione `ob_get_contents()`.
La sintassi della funzione è la seguente.

```
String ob_get_contents(void);
```

La tabella seguente descrive il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Restituzione di `ob_get_contents()`	`string`	La funzione restituisce il contenuto del buffer sotto forma di stringa.

Esempio di funzione:

```
$out = ob_get_contents();
```


Lo script che segue è una variante di quello precedente e mostra che è possibile accedere e visualizzare il contenuto del buffer prima di cancellarlo.

```
<?php

// Buffering dell'output - Esempio 25-4
//--------------------

ob_start();
set_error_handler("catchError");
echo("<h1>Benvenuti nel mondo del buffering.</h1>");
echo("Il buffering è utile per evitare la visualizzazione parziale di una
    pagina Web nel caso in cui si verifichi un errore.");

echo($doesnotexist);
ob_end_flush();

function catchError() {
    $out = ob_get_contents();
    ob_clean();
    echo("<br>Sfortunatamente si è verificato un errore!<br>");
    echo("È stato visualizzato '$out'");
}

?>
```

All'interno della funzione `catchError()` viene invocata la funzione `ob_get_contents()`, che memorizza il contenuto del buffer nella variabile `$out`. Questo contenuto verrà poi visualizzato. L'output di questo script è quello della Figura 25.3.

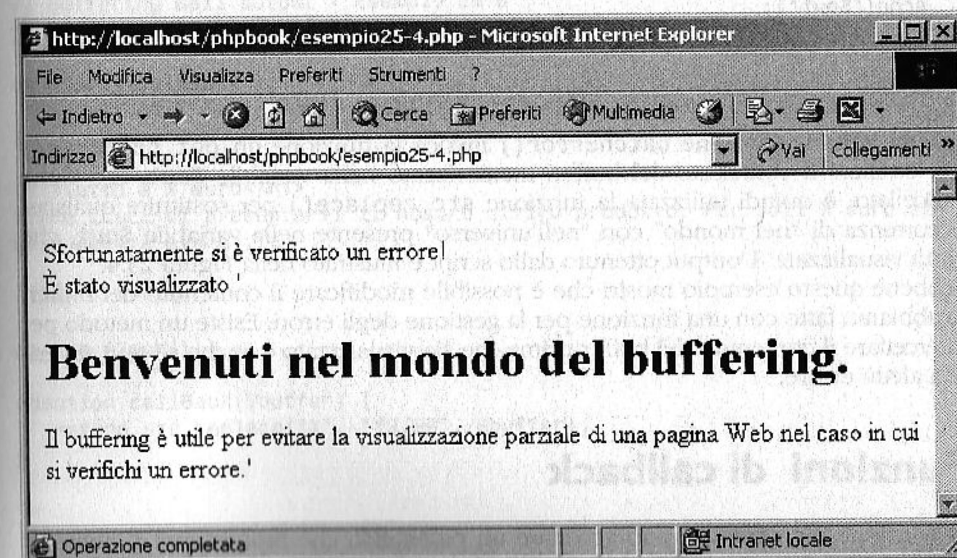


Figura 25.3
Visualizzazione dell'output del buffer.

La Figura 25.3 mostra che l'output preso dal buffer viene visualizzato dopo il messaggio d'errore.
Non è detto che questo possa essere particolarmente utile, tuttavia se potete accedere al contenuto del buffer, potete anche modificarlo.

Modifica dell'output

Lo script seguente mostra che la funzione `ob_get_contents()` permette di accedere al contenuto del buffer e modificarlo.

```
<?php

// Buffering dell'output - Esempio 25-5
//-------------------------

ob_start();
set_error_handler("catchError");
echo("<h1>Benvenuti nel mondo del buffering.</h1>");
echo("Il buffering è utile per evitare la visualizzazione parziale di una
      pagina Web nel caso in cui si verifichi un errore.");

echo($doesnotexist);
ob_end_flush();

function catchError() {
   $out = ob_get_contents();
   ob_clean();
   $out = str_replace("nel mondo", "nell'universo", $out);
   echo("$out");
}

?>
```

Questa volta la funzione `catchError()` invoca la funzione `ob_get_contents()` per ottenere il contenuto del buffer, memorizzato nella variabile `$out`, che viene cancellato; è quindi utilizzata la funzione `str_replace()` per sostituire qualsiasi occorrenza di "nel mondo" con "nell'universo" presente nella variabile `$out`, che verrà visualizzata. L'output ottenuto dallo script è illustrato nella Figura 25.4.
Sebbene questo esempio mostri che è possibile modificare il contenuto del buffer, lo abbiamo fatto con una funzione per la gestione degli errori. Esiste un metodo per intercettare il contenuto del buffer prima che sia visualizzato e anche se non si verifica alcun errore.

Funzioni di callback

La funzione `ob_start()` può accettare un parametro che rappresenta il nome di una funzione invocata appena prima della funzione `ob_end_flush()`.

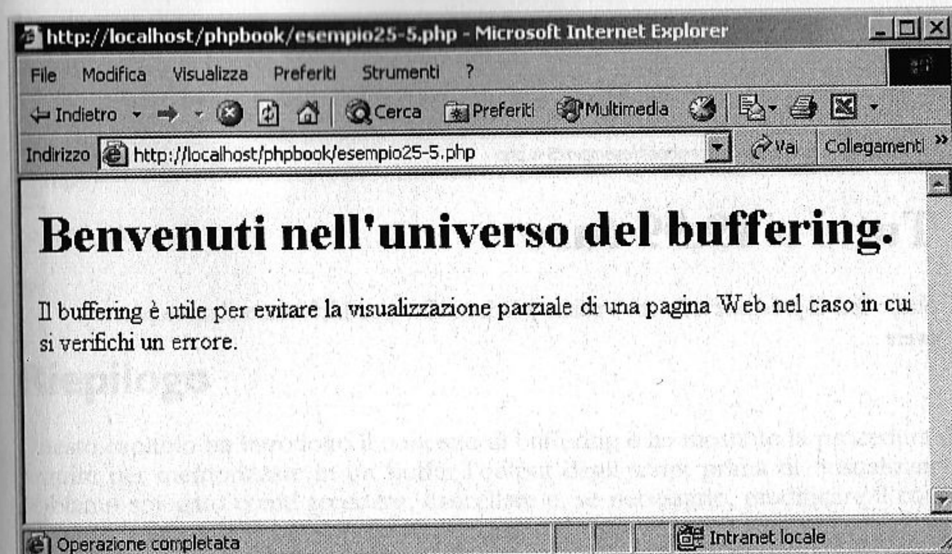


Figura 25.4
Modifica del contenuto del buffer.

La funzione invocata è nota con il nome di funzione di callback. Lo script seguente mostra l'esempio di una funzione simile.

```
<?php

// Buffering dell'output - Esempio 25-6
//----------------------

ob_start("callBack");

?>

<h1>Tutto a X euro</h1>
Siamo lieti di presentarvi il nostro ultimo prodotto. Per soli X euro al
mese potete avere ....

<?php

ob_end_flush();

function callBack($buffer) {
   return str_replace("X", "19,99", $buffer);
}

?>
```

La funzione, cui è stato assegnato il nome di `callBack()`, deve essere in grado di ricevere una stringa che accoglierà il contenuto del buffer. Inoltre dovrà restituire tale stringa in modo che il contenuto del buffer possa essere visualizzato nel browser. All'interno della funzione è possibile modificare il contenuto della stringa, come nell'esempio precedente. L'output ottenuto da questo script è quello della Figura 25.5.

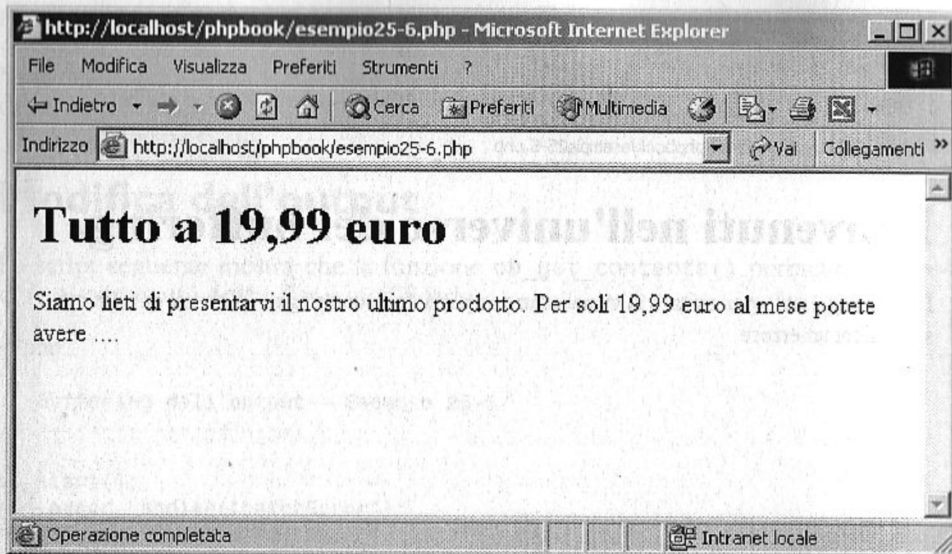


Figura 25.5
Modifica del contenuto di un buffer.

Compressione dell'output del buffer

Se nel leggere questo capitolo avete pensato che il buffer non possa esservi molto utile, considerate la funzione `ob_gzhandler()`.
La sintassi della funzione è la seguente.

```
String ob_gzhandler(string buffer)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
buffer	string	Contenuto del buffer.
Restituzione di `ob_gzhandler()`	string	Restituzione dell'output.

Questa funzione è progettata per essere una funzione di callback della funzione `ob_start()`. Essa riceve il contenuto del buffer e, se il browser Web è in grado di accettare pagine `gz_encoded`, per esempio pagine Web compresse, il contenuto del buffer viene compresso e inviato al browser.
Grazie a questa funzione, la velocità di trasmissione del sito Web aumenta e con essa aumenta anche la velocità di visualizzazione delle pagine Web. Di seguito è riportato un esempio dell'utilizzo di questa funzione in uno script.

```
<?php

// Buffering dell'output - Esempio 25-7
//---------------------

ob_start("ob_gzhandler");
```



```
?>

<h1>Pagina Web rapida</h1>
Questa pagina sarà compressa e inviata al browser se quest'ultimo supporta
la compressione.

<?php

ob_end_flush();

?>
```

Riepilogo

Questo capitolo ha introdotto il concetto di buffering e ha mostrato la procedura da seguire per memorizzare in un buffer l'output degli script prima di visualizzarlo. Abbiamo spiegato come accedere, cancellare e, se necessario, modificare il contenuto del buffer. Abbiamo concluso introducendo il concetto di funzioni di callback e, in particolare, ci siamo occupati della funzione `ob_gzhandler()`, che può velocizzare gli accessi alle pagine Web. Il prossimo capitolo analizzerà il ruolo dei file in PHP e mostrerà come crearli e manipolarli.

Parte VII

Gestione dei file

Capitolo 26

Lettura e scrittura su file

Introduzione

In questo capitolo si vedrà come leggere dati da file e scrivere informazioni su di essi. In termini fisici, un file è una collezione sequenziale di caratteri che possono essere manipolati. In termini logici, un file può essere visto come una collezione di record di dati oppure come un programma per computer. Molti sistemi informatici utilizzano i file per archiviare le informazioni in modo da potervi accedere in futuro. PHP offre alcuni strumenti per utilizzare e manipolare i file.

Apertura di un file

PHP può aprire un file ricorrendo a una funzione molto semplice, denominata `fopen()`. La sintassi della funzione è la seguente.

```
int fopen (string nomefile, string modalità);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomefile	string	Il nome del file e il suo percorso completo.
modalità	string	"r" apre in sola lettura; colloca il puntatore al file all'inizio del file. "r+" apre in lettura e scrittura; colloca il puntatore al file all'inizio del file. "w" apre in sola scrittura; colloca il puntatore al file all'inizio del file e tronca il file alla lunghezza zero. Se il file non esiste, tenta di crearlo. "w+" apre in lettura e scrittura; colloca il puntatore al file all'inizio del file e tronca il file alla lunghezza zero. Se il file non esiste, tenta di crearlo. "a" apre in sola scrittura; colloca il puntatore al file alla fine del file. Se il file non esiste, tenta di crearlo. "a+" apre in lettura e scrittura; colloca il puntatore al file alla fine del file. Se il file non esiste, tenta di crearlo.
Restituzione di `fopen()`	int	Una connessione stream al file specificato.

La tabella precedente illustra le varie modalità in cui è possibile aprire un file.
Esempi di funzioni:

```
$fp = fopen ("/home/book/file.txt", "r");
$fp = fopen ("/home/book/file.gif", "wb");
$fp = fopen ("http://www.example.com/file.txt", "r+");
$fp = fopen ("ftp://user:password@example.com/file.txt", "w");
```

PHP consente di aprire un file sulla macchina locale, nonché su un computer remoto (se le autorizzazioni di protezione lo consentono). L'accesso ai file remoti può avvenire con una connessione HTTP o FTP. Se il nome del file inizia con http://, viene aperta una connessione HTTP con il server specificato e viene restituito un puntatore al file richiesto. Se il nome del file inizia con ftp://, viene aperta una connessione FTP con il server specificato e viene restituito un puntatore al file richiesto.
Se il nome del file inizia con qualunque altra cosa, esso verrà aperto dal sistema locale e verrà restituito un puntatore al file richiesto. Se la funzione fallisce, viene restituito FALSE. I file possono essere aperti in molte modalità diverse in relazione a ciò che volete farne.

Chiusura di un file

Una volta ultimato l'accesso a un file, dovete chiuderlo per liberare la memoria e informare il sistema operativo che avete finito. In PHP viene utilizzata la funzione `fclose()` per chiudere un file. Il puntatore al file dev'essere valido e deve puntare a un file aperto con `fopen()`.
La sintassi della funzione è la seguente.

```
bool fclose (int puntatoreAFile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatoreAFile	int	Il puntatore a file prodotto da `fopen()`.
Restituzione di `fclose()`	boolean	TRUE in caso di successo e FALSE in caso contrario.

Esempio di funzione:

```
$fp = fopen ("/home/book/file.txt", "r"); fclose($fp);
```

Ottenere la dimensione di un file

La dimensione di un file in byte è un'informazione che può essere ottenuta facilmente con la funzione `filesize()`.
La sintassi della funzione è la seguente.

```
int filesize (string nomefile);
```


La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomefile	string	Il file da utilizzare e il suo percorso completo.
Restituzione di `filesize()`	int	Restituisce la dimensione del file in byte.

Esempio di funzione:

```
$bytes = filesize ($filename);
```

Questa funzione non è utilizzabile con i file remoti e il file da esaminare dev'essere accessibile attraverso il filesystem del server.

Lettura di un intero file

Dopo aver aperto un file è possibile leggerne il contenuto in tutto o in parte. Per leggere l'intero file (dall'inizio alla fine), è possibile utilizzare la funzione `fread()`.
La sintassi della funzione è la seguente.

```
string fread (int puntatoreAFile, int lunghezza);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatoreAFile	int	Il puntatore a file prodotto da `fopen()`.
lunghezza	int	La dimensione totale del file.
Restituzione di `fread()`	string	Intero file come testo in una stringa.

Esempi di funzioni:

```
$fp = fopen ($filename, "r");
$contents = fread ($fp, filesize ($filename));
```

Notate che è necessaria la funzione `filesize()` per determinare la dimensione del file.

Lettura di un intero file in una stringa

Un'altra funzione che è possibile utilizzare per leggere il file intero è `file_get_contents()`.
La sintassi della funzione è la seguente.

```
string file_get_contents ( string nomefile );
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomefile	string	Nome del file.
Restituzione di file_get_contents()	string	Contenuto del file.

Esempi di funzioni:

```
$contents = file_get_contents ("test.txt");
$contents = file_get_contents ("http://www.yahoo.com");
```

Con questa funzione, il contenuto del file viene collocato in una stringa. È possibile utilizzare un URL come nome del file con questa funzione se i wrapper `fopen()` sono stati attivati. Se un file è troppo grande, questa funzione non è adatta.

Lettura di un intero file in un array

Esiste una funzione chiamata `file()` che viene utilizzata per leggere un intero file in un array, dove ogni elemento dell'array corrisponde a una riga nel file. Se il file è di dodici righe, l'array prodotto conterrà dodici elementi.
La sintassi della funzione è la seguente.

```
array file ( string nomefile );
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomefile	string	Nome del file da utilizzare.
Restituzione di file()	array	Inserisce il file in un array.

Esempi di funzioni:

```
$content = file("test.txt");
$line1 = $content[0];
$line2 = $content[1];
```

Lettura di un carattere da un file

La lettura dei caratteri da un file può avvenire con la funzione `fgetc()`, che ha la sintassi seguente.

```
string fgetc(int puntatoreAFile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatoreAFile	int	Il puntatore a file prodotto da fopen().
Output di fgetc()	string	Stringa contenente un singolo carattere letto dal file.

Esempi di funzioni:

```
$fp = fopen ($filename, "r");
$character = fgetc($fp);
```

La funzione richiede un solo parametro, il puntatore a file, che viene fornito dalla funzione `fopen()`, e restituisce una stringa contenente un singolo carattere letto dal file. Lo script che segue visualizza il primo carattere del file test.txt, un semplice file di testo memorizzato nella sottodirectory *files* e contenente il seguente testo:

```
1. Sopra la panca la capra campa, sotto la panca la capra crepa.
2. Sopra la panca la capra campa, sotto la panca la capra crepa.
3. Sopra la panca la capra campa, sotto la panca la capra crepa.
4. Sopra la panca la capra campa, sotto la panca la capra crepa.
```

Questo script di esempio può essere utilizzato con qualunque file per rilevare il primo carattere.

```
<?php

// Lettura e scrittura su file - Esempio 26-1
//-------------------------------

$fp1 = fopen("files/test.txt", "r");
$c = fgetc($fp1);

echo("Il primo carattere è $c");

fclose($fp1);

?>
```

Lettura di un file una riga alla volta

Oltre a leggere i singoli caratteri da un file, PHP supporta anche la lettura di un'intera riga di caratteri. Questo è possibile grazie alla funzione `fgets()`, che ha la sintassi seguente.

```
string fgets (int puntatoreAFile, int lunghezza);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatoreAFile	int	Il puntatore a file prodotto da fopen().
lunghezza	int	La dimensione totale del file.
Restituzione di fgets()	string	Restituisce una stringa, che può essere al massimo di lunghezza –1 byte, letta dal file cui punta *puntatoreAFile*.

Esempi di funzioni:

```
$s = fgets($fp1, 4096);
```

La funzione leggerà un certo numero di caratteri in una stringa fino al raggiungimento del valore *lunghezza*, alla lettura di un carattere newline o al raggiungimento del marcatore di file del file. Per rilevare il marcatore di file del file è necessaria la funzione `feof()`.
La sintassi della funzione è la seguente.

```
int feof (int puntatoreAFile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatoreAFile	int	Il puntatore a file prodotto da `fopen()`.
Restituzione di `feof()`	int	Restituisce TRUE se è alla fine del file, altrimenti restituisce FALSE.

Esempi di funzioni:

```
$s = feof($fp1);
```

Lo script che segue illustra un esempio di lettura di un file riga per riga.

```
<?php

// Lettura e scrittura su file - Esempio 26-2
//------------------------------

$fp1 = fopen("files/test.txt", "r");

echo("Il contenuto del file test.txt è:<p>");
echo("<form>");
echo("<textarea cols=70 rows=15>");
$c=1;
$s = fgets($fp1, 4096);

while(!feof($fp1)) {
      echo("$c: $s");
      $c++;
      $s = fgets($fp1, 4096);
}
echo("</textarea></form>");
fclose($fp1);

?>
```

Lettura di un file remoto

L'apertura di un file remoto può avvenire con la stessa facilità di un file locale, purché si disponga delle autorizzazioni necessarie. Lo script che segue consente di specificare il nome di una pagina Web (che ovviamente è memorizzata come file), che viene poi letta e visualizzata in un campo textarea.


```
<?php

// Lettura e scrittura su file - Esempio 26-3
//---------------------------------

if (isset($_POST['webpage']))
      $webpage = $_POST['webpage'];
else
      $webpage = "";

echo "Inserire l'indirizzo della pagina Web http://";
echo "<form action='" . $_SERVER['PHP_SELF'] ."' method='post'>";
echo "<input type='text' size='30' name='webpage' value='$webpage'>";
echo "<input type = 'submit' value='Visualizza'>";
echo "</form>";

if ($webpage) {
   $fp1 = fopen("http://" . $_POST['webpage'],"r");
   echo "Il contenuto della pagina Web è:<p>";
   echo "<form>";
   echo "<textarea cols=70 rows=15>";
   $c=1;
   $s = fgets($fp1, 4096);
   while(!feof($fp1)) {
     echo "$s";
       $s = fgets($fp1, 4096);
   }
   echo "</textarea></form>";
   fclose($fp1);
}

?>
```

L'output di questo script è quello della Figura 26.1.

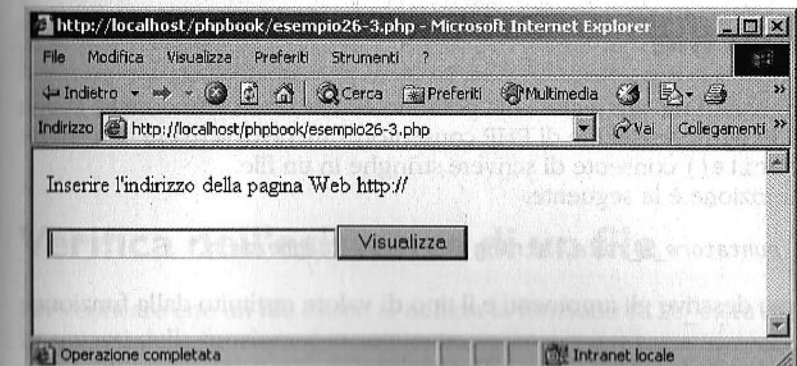


Figura 26.1
Lettura di un file remoto.

Se viene fornito l'indirizzo di una pagina Web, il suo contenuto viene collocato in un campo textarea, come mostrato nella Figura 26.2.

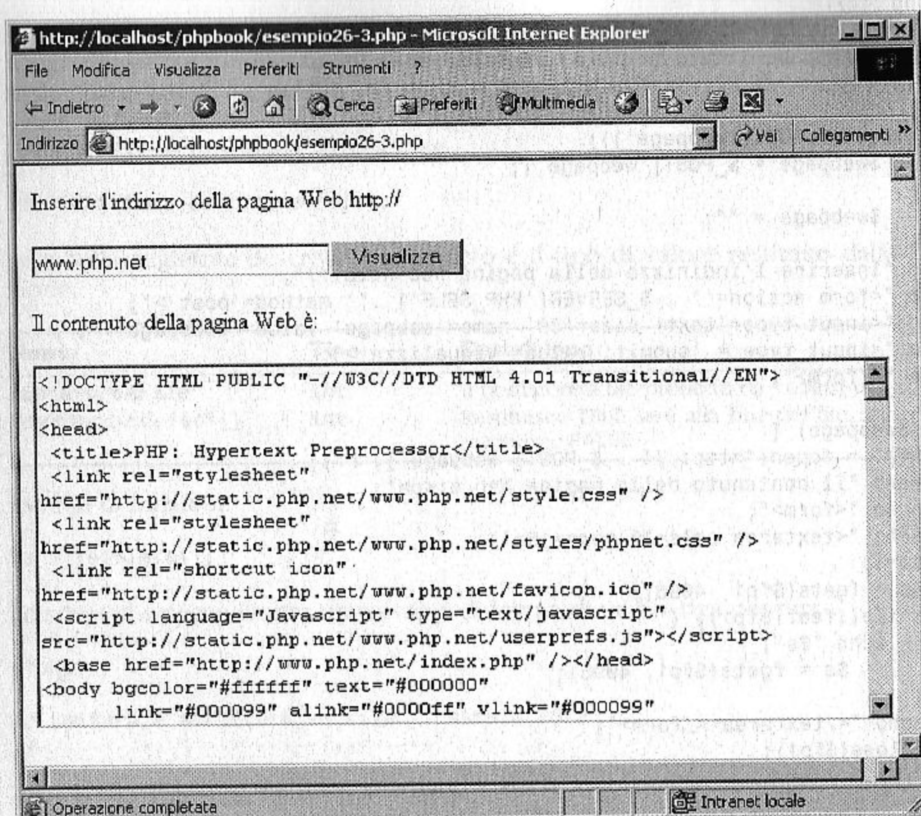


Figura 26.2
Risultato della lettura di un file remoto.

Scrittura su file

Finora gli esempi hanno mostrato che i file possono essere letti e visualizzati. Tuttavia, gli strumenti di gestione dei file di PHP consentono anche di scrivere dati sui file. La funzione `fwrite()` consente di scrivere stringhe in un file.
La sintassi della funzione è la seguente.

```
int fwrite(int puntatore_a_file, string str, int lunghezza);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
puntatore_a_file	int	Il puntatore a file prodotto da `fopen()`.
lunghezza	int	La dimensione totale del file che legge.
Restituzione di `fwrite()`	int	Numero di byte scritti nel file.

Esempi di funzioni:

```
$fp = fopen ($filename, "w");
$fw = fwrite ($fp, $content );
```

L'esempio che segue utilizza la funzione `fopen()` per creare un file "test2.txt" (con l'opzione `"w"`) e `fwrite()` per scrivervi una stringa. Il nuovo file verrà creato nella directory *files* e deve avere le autorizzazioni corrette, in modo che PHP possa scrivere in esso.

```
<?php

// Lettura e scrittura su file - Esempio 26-4
//-----------------------------------

$filename = "files/test2.txt";
$content = "Ciao, questo è il mio file.";

$fp = fopen($filename, "w");

$fw = fwrite($fp, $content);

fclose($fp);

?>
```

Esaminando il contenuto della directory *files* e visualizzando il file test2.txt si ottiene il file della Figura 26.3.

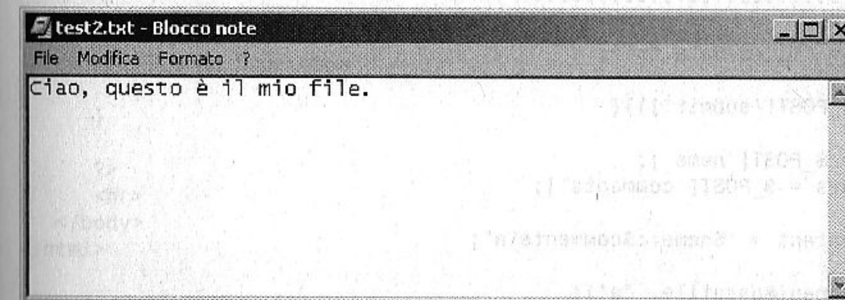


Figura 26.3
Il file creato.

Verifica dell'esistenza di un file

Per verificare che un file esista, si utilizza la funzione `file_exists()`.
La sintassi della funzione è la seguente.

```
Bool file_exists(string nomefile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomefile	string	Nome del file.
Restituzione di file_exists()	boolean	Restituisce TRUE se il file esiste o FALSE in caso contrario.

Esempio di funzione:

```
$file = file_exists($filename);
```

Realizzazione di un guestbook con le funzioni di manipolazione dei file

Un'applicazione molto comune che sfrutta le funzioni di manipolazione dei file è il guestbook, una pagina utilizzata da molti siti Web per consentire agli utenti di lasciare il proprio nome e qualche commento. Con le funzioni di lettura e scrittura è possibile rispettivamente leggere i messaggi memorizzati e scriverne di nuovi. L'esempio che segue introduce un modo rapido per implementare un'applicazione di questo tipo.

```
<?php

// Lettura e scrittura su file - Esempio 26-5
//---------------------------------

$datafile = "files/data.txt";

if(isset($_POST['submit'])){

   $name = $_POST['name'];
   $comments = $_POST['comments'];

   $new_content = "$name::$comments\n";

   $fp = fopen($datafile, "a");
   $fw = fwrite($fp, $new_content);
   $fc = fclose($fp);
}

?>

<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <b>GUESTBOOK </b>
      <form name="form1" method="post"
        action="<?php echo $_SERVER['PHP_SELF'];?>">

      <table width="30%" border="0" cellspacing="0" cellpadding="0"
        height="50">
```



```
         <tr>
            <td width="31%">Nome:</td>
            <td width="69%"><input type="text" name="name"></td>
         </tr>

         <tr>
            <td width="31%">Commenti:</td>
            <td width="69%"><textarea name="comments"
              rows="3"></textarea></td>
         </tr>

      </table>

      <input type="submit" name="submit" value="Invia commento">
      <input type="reset" name="Reset" value="Reimposta">
      <br>

      </form>
      <hr>
      <?php

      //Lettura di dati da un file e creazione di un ciclo per la loro
      //visualizzazione

      if (file_exists($datafile)) {
         $read_data = file($datafile);
         $num_comments = count($read_data);

         for($c=0;$c<$num_comments-1;$c++){
            $content = explode("::", $read_data[$c]);
            echo "<b>Nome:</b> $content[0]";
            echo "<br>";
            echo "<b>Commento:</b> $content[1]";
            echo "<br><br>";
         }
      }

      ?>
      <hr>
   </body>
</html>
```

La prima parte dello script elabora i dati del modulo e apre il file.

```
$$datafile = "files/data.txt";

if(isset($_POST['submit'])){

   $name = $_POST['name'];
   $comments = $_POST['comments'];

   $new_content = "$name::$comments\n";

   $fp = fopen($datafile, "a");
   $fw = fwrite($fp, $new_content);
   $fc = fclose($fp);
}
```

La parte successiva visualizza il modulo:

```
<html>
   <body bgcolor="#FFFFFF" text="#000000">
      <b>GUESTBOOK </b>
      <form name="form1" method="post"
        action="<?php echo $_SERVER['PHP_SELF'];?>">

      <table width="30%" border="0" cellspacing="0" cellpadding="0"
       height="50">

         <tr>
            <td width="31%">Nome:</td>
            <td width="69%"><input type="text" name="name"></td>
         </tr>

         <tr>
            <td width="31%">Commenti:</td>
            <td width="69%"><textarea name="comments"
                  rows="3"></textarea></td>
         </tr>
       </table>

       <input type="submit" name="submit" value="Invia commento">
       <input type="reset" name="Reset" value="Reimposta">
       <br>

       </form>
       <hr>
```

La parte finale verifica se il file esiste e visualizza il contenuto del guestbook:

```
    if (file_exists($datafile)) {
       $read_data = file($datafile);
       $num_comments = count($read_data);

       for($c=0;$c<$num_comments-1;$c++){
          $content = explode("::", $read_data[$c]);
          echo "<b>Nome:</b> $content[0]";
          echo "<br>";
          echo "<b>Commento:</b> $content[1]";
          echo "<br><br>";
       }
    }
```

La Figura 26.4 illustra l'output prodotto dallo script precedente.

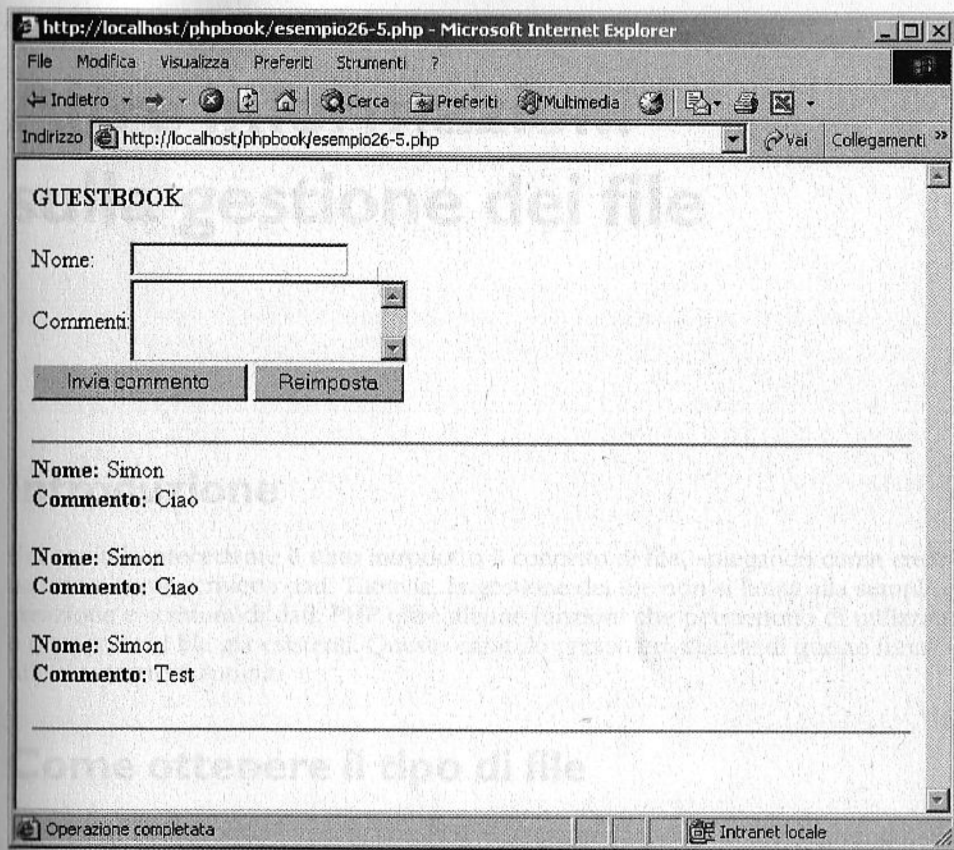


Figura 26.4
Output del guestbook.

Riepilogo

Questo capitolo ha illustrato come compiere alcune semplici operazioni di manipolazione dei file. Sono state introdotte le funzioni che permettono di leggere e scrivere su file esistenti o che devono essere creati. Il capitolo si è chiuso con la creazione di un semplice script per la realizzazione di un guestbook. Nel prossimo capitolo verranno esaminati in modo più approfondito alcuni altri strumenti di gestione dei file forniti da PHP.

Altre informazioni sulla gestione dei file

Introduzione

Nel capitolo precedente è stato introdotto il concetto di file, spiegando come crearne uno e come scrivervi dati. Tuttavia, la gestione dei file non si limita alla semplice creazione e scrittura di dati. PHP offre alcune funzioni che permettono di utilizzare e manipolare i file già esistenti. Questo capitolo presenterà alcune di queste funzioni e ne mostrerà l'utilità.

Come ottenere il tipo di file

Avete già avuto modo di vedere che PHP è in grado di aprire un file ricorrendo alla funzione `fopen()`.
A volte può essere utile sapere se si ha a che fare con un file normale o una directory. PHP mette a disposizione due funzioni che aiutano a stabilire il tipo di file. La sintassi delle funzioni è riportata di seguito.

```
bool is_file (string nomefile);
bool is_dir (string nomefile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
nomefile	string	Il nome del file e il relativo percorso di sistema completo.
Restituzione di `is_file()`	boolean	TRUE se il nome di file specificato esiste ed è un file normale.
Restituzione di `is_dir()`	boolean	TRUE se il nome di file specificato esiste ed è una directory.

Esempi di funzioni:

```
$checkFile = is_file ("/home/book/file.txt");
$checkDir = is_dir ("/home/book/");
```

Queste funzioni non agiscono sui file remoti, quindi il nome di file da esaminare dev'essere accessibile tramite il filesystem del server. Lo script che segue illustra l'utilizzo di queste funzioni.

```
<?php

// Gestione dei file - Esempio 27-1
//------------------------------------------------

$datafile = "files/data.txt";

if(isset($_POST['submit'])){
   $filename = $_POST['filename'];
   if (is_file($filename))
      echo "$filename è un file";
   elseif (is_dir($filename))
      echo "$filename è una directory";
   else
      echo "Non so cosa sia $filename!";
}

?>
<html>
   <body bgcolor="#FFFFFF" text="#000000">
   <h2>Controllo dei file</h2>

   <form name="form1" method="post"
    action="<?php echo $_SERVER['PHP_SELF'];?>">

      Nome del file:<input type="text" name="filename">
      <input type="submit" name="submit" value="Invia nome file">
      <input type="reset" name="Reset" value="Reimposta">

   </form>

   </body>
</html>
```

Lo script è costituito da due parti principali. La prima stabilisce se il modulo è stato inviato, poi accede al nome di file:

```
$datafile = "files/data.txt";

if(isset($_POST['submit'])){
   $filename = $_POST['filename'];
```

successivamente invoca le funzioni `is_file()` e `is_dir()` per stabilire se il file indicato è un file o una directory, ma può anche non riuscire a stabilire di che cosa si tratti:

```
   if (is_file($filename))
      echo "$filename è un file";
   elseif (is_dir($filename))
      echo "$filename è una directory";
   else
      echo "Non so cosa sia $filename!";
}
```


L'ultima parte dello script visualizza il modulo che l'utente utilizza per immettere il nome del file:

```
<html>
   <body bgcolor="#FFFFFF" text="#000000">
   <h2>Controllo dei file</h2>

   <form name="form1" method="post"
    action="<?php echo $_SERVER['PHP_SELF'];?>">

      Nome del file:<input type="text" name="filename">
      <input type="submit" name="submit" value="Invia nome file">
      <input type="reset" name="Reset" value="Reimposta">

   </form>

   </body>
</html>
```

L'output ottenuto da questo script è quello della Figura 27.1.

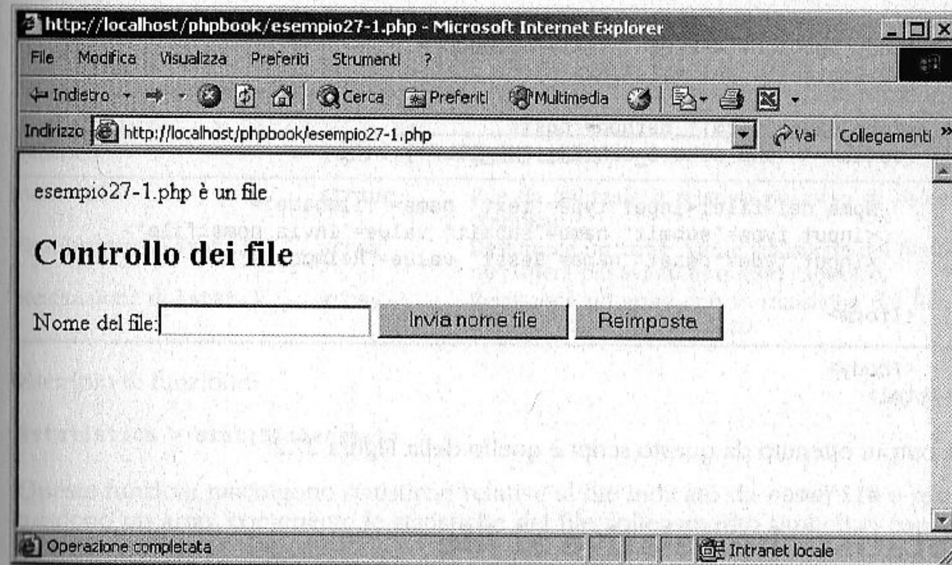


Figura 27.1
File e directory.

Dimensioni dei file

Alcuni file possono raggiungere dimensioni notevoli, quindi è molto importante poter controllare la dimensione di un file prima di cominciare a utilizzarlo. Dal capitolo precedente sappiamo che è possibile ricorrere alla funzione `filesize()` per de-

terminare la dimensione di un file. Di seguito è riportata una nuova versione dello script precedente che visualizza la dimensione del file espressa in byte.

```
<?php

// Gestione dei file - Esempio 27-2
//-------------------------------------------------

$datafile = "files/data.txt";

if(isset($_POST['submit'])){
  $filename = $_POST['filename'];
  if (is_file($filename)) {
    $bytes = filesize($filename);
    echo "$filename è un file di $bytes byte.";
  }
  else
    echo "Non so cosa sia $filename!";
}

?>

<html>
  <body bgcolor="#FFFFFF" text="#000000">
  <h2>Controllo dei file</h2>

  <form name="form1" method="post"
   action="<?php echo $_SERVER['PHP_SELF'];?>">

    Nome del file:<input type="text" name="filename">
    <input type="submit" name="submit" value="Invia nome file">
    <input type="reset" name="Reset" value="Reimposta">

  </form>

  </body>
</html>
```

L'output ottenuto da questo script è quello della Figura 27.2.

Statistiche relative al file

Con le funzioni `stat()` e `lstat()` si possono ottenere ulteriori informazioni su un file. La differenza principale tra queste due funzioni è che, oltre alle informazioni sui file, `lstat()` è in grado di restituire anche informazioni sui collegamenti simbolici. Queste funzioni non agiscono sui file remoti o sulle cartelle locali, pertanto il file da esaminare dev'essere accessibile tramite il filesystem del server.
La sintassi delle funzioni è riportata di seguito.

```
array stat (string nomefile);
array lstat (string nomefile);
```

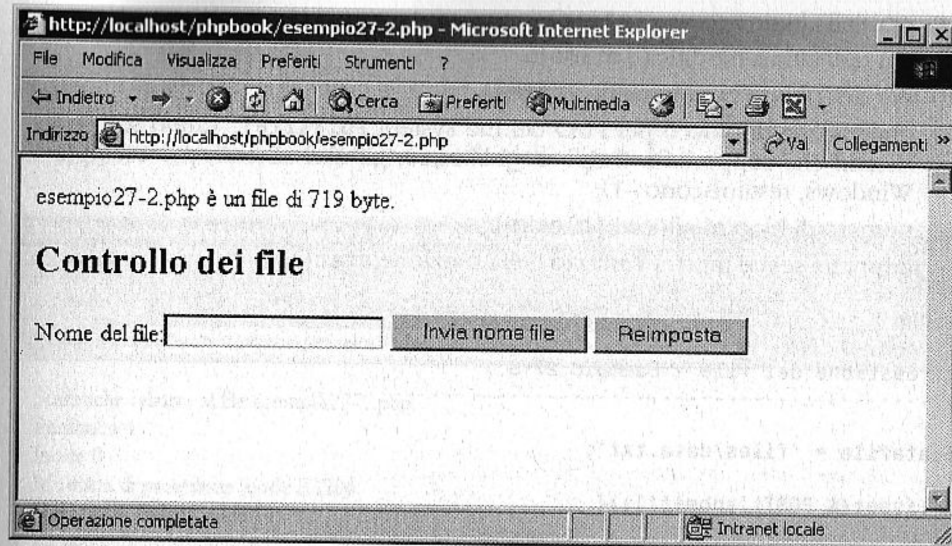



Figura 27.2
Dimensione del file espressa in byte.

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
nomefile	string	File da utilizzare e relativo percorso di sistema completo.
Restituzione di `stat()`	array	Restituisce un array con le statistiche del file (se ne parlerà più avanti in questo capitolo).
Restituzione di `lstat()`	array	Restituisce un array con le statistiche del file o del collegamento simbolico.

Esempio di funzione:

```
$statistics = stat($filename);
```

Queste funzioni raccolgono statistiche relative al file indicato da *`nomefile`* e restituiscono un array contenente le statistiche del file/collegamento simbolico con gli elementi indicati di seguito.

0. periferica `[dev]`;
1. inode `[ino]`;
2. modalità di protezione inode `[mode]`;
3. numero di collegamenti `[nlink]`;
4. ID utente del proprietario `[uid]`;
5. ID di gruppo del proprietario `[gid]`;
6. tipo di periferica se periferica inode `[rdev]` (valido solo sui sistemi che supportano il tipo `st_blksize`; gli altri sistemi (per esempio Windows) restituiscono –1);
7. dimensione in byte `[size]`;

8. ora dell'ultimo accesso [atime];
9. ora dell'ultima modifica [mtime];
10. ora dell'ultimo cambiamento [ctime];
11. dimensione di blocco per l'I/O del file system [blksize] (valido solo sui sistemi che supportano il tipo st_blksize; gli altri sistemi, per esempio Windows, restituiscono –1);
12. numero di blocchi allocati [blocks];

Lo script che segue mostra l'utilizzo della funzione stat().

```
<?php

// Gestione dei file - Esempio 27-3
//-----------------------------------------------

$datafile = "files/data.txt";

if(isset($_POST['submit'])){

    $filename = $_POST['filename'];

if (is_file($filename)) {
   $stats = stat($filename);
   echo "Statistiche relative al file $filename:";
   echo "<br>Periferica " . $stats['dev'];
   echobr>Inode " . $stats['ino'];
   echo "<br>Modalità di protezione inode " . $stats['mode'];
   echo "<br>Numero di collegamenti " . $stats['nlink'];
   echo "<br>ID utente del proprietario " . $stats['uid'];
   echo "<br>ID di gruppo del proprietario " . $stats['gid'];
   echo "<br>Tipo di periferica se periferica inode " . $stats['rdev'];
   echo "<br>Dimensione in byte " . $stats['size'];
   echo "<br>ora dell'ultimo accesso " . $stats['atime'];
   echo "<br>Ora dell'ultima modifica " . $stats['mtime'];
   echo "<br>Ora dell'ultimo cambiamento " . $stats['ctime'];
   echo "<br>Dimensione di blocco per l'I/O del file
      system " .$stats['blksize'];
   echo "<br>Numero di blocchi allocati " . $stats['blocks'];

else
    echo "Non so cosa sia $filename is!";
}

?>
<html>
  <body bgcolor="#FFFFFF" text="#000000">
    <h2>Statistiche sui file</h2>

    <form name="form1" method="post" action="<?php echo
      $_SERVER['PHP_SELF'];?>">

    Nome del file:<input type="text" name="filename">
    <input type="submit" name="submit" value="Invia nome file">
    <input type="reset" name="Reset" value="Reimposta">
```



```
    </form>

  </body>
</html>
```

L'output ottenuto da questo script è quello della Figura 27.3.

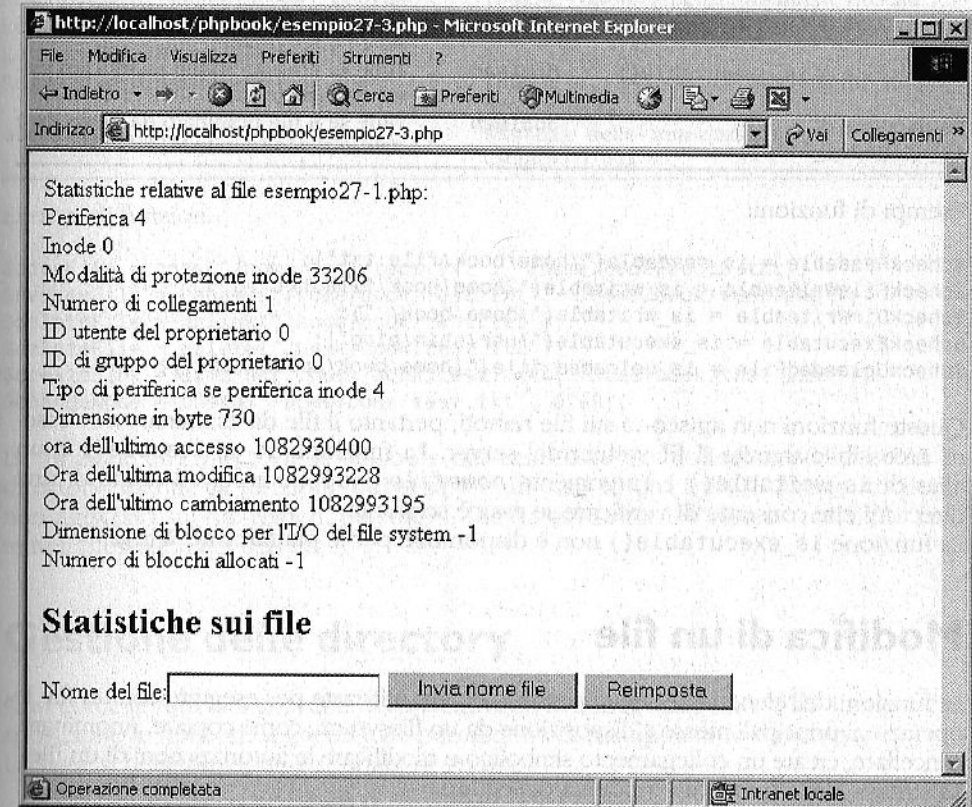


Figura 27.3
Statistiche relative alle dimensioni in byte.

Verifica degli attributi dei file

Le quattro funzioni di cui all'elenco che segue consentono di esaminare le proprietà degli attributi dei file. Tali funzioni permettono infatti di verificare facilmente se un file è leggibile, scrivibile, eseguibile o caricato.
La sintassi delle funzioni è riportata di seguito.

```
bool is_readable(string nomefile);
bool is_writable(string nomefile);
bool is_executable(string nomefile);
bool is_uploaded_file(string nomefile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
nomefile	string	Il nome del file e il relativo percorso di sistema completo.
Restituzione di `is_readable()`	boolean	`TRUE` se il nome di file specificato esiste ed è leggibile.
Restituzione di `is_writable()`	boolean	`TRUE` se il nome di file specificato esiste ed è scrivibile.
Restituzione di `is_executable()`	boolean	`TRUE` se il nome di file specificato esiste ed è eseguibile.
Restituzione di `is_uploaded_file()`	boolean	`TRUE` se il file nominato da filename è stato caricato tramite HTTP POST.

Esempi di funzioni:

```
$checkReadable = is_readable("/home/book/file.txt");
$checkFileWriteable = is_writable("/home/book/file.txt");
$checkDirWriteable = is_writable("/home/book/ ");
$checkExecutable = is_executable("/usr/sbin/ping");
$checkUploadedFile = is_uploaded_file("/home/book/upload.txt");
```

Queste funzioni non agiscono sui file remoti, pertanto il file da esaminare dev'essere accessibile tramite il filesystem del server. La funzione `is_writeable()` è un alias di `is_writable()` e l'argomento *`nomefile`* potrebbe essere il nome di una directory che consente di verificare se essa è scrivibile.
La funzione `is_executable()` non è disponibile per le piattaforme Win32.

Modifica di un file

Le funzioni dell'elenco che segue possono essere utilizzate per eseguire con un file le operazioni principali messe a disposizione da un filesystem, come copiare, rinominare, cancellare, creare un collegamento simbolico e modificare le autorizzazioni di un file. La sintassi delle funzioni è riportata di seguito.

```
int copy(string origine, string destinazione);
bool rename(string vecchionome, string nuovonome);
int unlink(string nomefile);
int delete(string nomefile);
int symlink(string destinazione, string collegamento);
int chmod(string nomefile, int modalità);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
nomefile	string	Il nome del file e il relativo percorso di sistema completo.
origine	boolean	Il nome del file di origine.
destinazione	boolean	Il nuovo nome del file, una copia grezza del file di origine.
vecchionome	string	Il nome corrente del file.
nuovonome	string	Il nuovo nome del file.

Nome	Tipo	Descrizione
destinazione	string	Il nome del file di destinazione del collegamento.
collegamento	string	Il nome del collegamento.
Restituzione di `copy()`	int	`TRUE` se l'operazione di copia ha avuto esito positivo.
Restituzione di `rename()`	boolean	`TRUE` se l'operazione di rinomina è riuscita.
Restituzione di `unlink()`	int	`TRUE` se l'operazione è riuscita.
Restituzione di `delete()`		
Restituzione di `symlink()`	int	`TRUE` se la creazione del collegamento simbolico è riuscita.
modalità	int	Modalità delle autorizzazioni del file (per esempio 0755).

Esempi di funzioni:

```
$copyFile = copy("/home/book/test.txt", "/home/book/test2.txt");
$renameFile = rename("/home/book/test.txt", "/home/book/test.php");
$deleteFile = unlink("/home/book/test.txt");
$deleteFile = delete("/home/book/test.txt");
$createLink = symlink("/home/book/test.txt","/home/book/test.link");
$changeMode = chmod("/home/book/test.txt", 0755);
```

La funzione `unlink()` viene utilizzata per cancellare file, collegamenti simbolici e directory, mentre la funzione `delete()` è un alias di `unlink()`. Queste funzioni non agiscono sui file remoti, pertanto il file da utilizzare dev'essere accessibile tramite il filesystem del server.

Gestione delle directory

Così come le funzioni precedenti consentivano di manipolare i file, PHP mette a disposizione le funzioni seguenti, che permettono di manipolare le directory.
La sintassi delle funzioni è riportata di seguito.

```
int mkdir(string percorso, int modalità);
bool rmdir(string nomedir);
bool rename(string vecchionome, string nuovonome);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
percorso	string	Percorso reale dove sarà creata la directory.
modalità	int	Modalità di autorizzazione del file (per esempio 755).
nomedir	string	Directory da eliminare.
vecchionome	string	Nome corrente del file.
nuovonome	string	Nuovo nome del file.
Restituzione di `mkdir()`	int	Restituisce 1 in caso di successo, 0 in caso contrario.
Restituzione di `rmdir()`	boolean	Restituisce `TRUE` in caso di successo, `FALSE` in caso contrario.
Restituzione di `rename()`	boolean	Restituisce `TRUE` in caso di successo, `FALSE` in caso contrario.

Esempi di funzioni:

```
$createDir = mkdir("/home/book/new", "0755");
$rmDir = rmdir("/home/book/new");
$renameDir = rename("/home/book/old", "/home/book/new);
```

Quando si crea una nuova directory con `mkdir()`, è possibile assegnare le modalità per le autorizzazioni con il parametro *`modalità`*. Si dovrebbe inoltre specificare il percorso originale, mentre l'ubicazione in cui si creerà la nuova directory dovrebbe avere le autorizzazioni di scrittura. Quando si utilizza `rmdir()`, la directory specificata dovrebbe essere vuota e le autorizzazioni più importanti devono consentire l'operazione.

Apertura e lettura di una directory

Esistono tre funzioni molto potenti che danno la possibilità di aprire una directory nel filesystem e di leggerne il contenuto prima di chiuderla. In questo modo possiamo vedere (e, volendo, visualizzare) il contenuto della directory.
La sintassi delle funzioni è riportata di seguito.

```
resource opendir(string percorso);
string readdir(resource dirHandle)
void closedir(resource dirHandle)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	**Tipo**	**Descrizione**
`percorso`	`string`	La directory da utilizzare con il percorso di sistema completo
`dirHandle`	`resource`	Handle della directory ottenuto da `opendir()`.
Restituzione di `opendir()`	`resource`	Handle della directory.
Restituzione di `readdir()`	`string`	Nome(i) del file successivo tratto dalla directory.
Restituzione di `closedir()`	`void`	Non restituisce alcunché.

Esempi di funzioni:

```
$openDirectory = opendir("/home/book/new");
$readDirectory = readdir($openDir);
$closeDirectory = closedir($openDir);
```

Se "percorso" non è una directory valida, o se la directory non può essere aperta per limitazioni nelle autorizzazioni o per errori nel filesystem, `opendir()` restituisce `FALSE` e genera un errore PHP. Per prevenire errori simili, inserite un simbolo `@` davanti a ciascuna funzione in modo da impedire che l'errore venga visualizzato nel browser (per esempio, `$open = @opendir(..);`). Questa soluzione è stata citata per la prima volta nel Capitolo 24. Lo script seguente mostra un esempio di utilizzo di queste funzioni.

```
<?php

// Gestione dei file - Esempio 27-4
//----------------------------------------
$datafile = "files/data.txt";

if(isset($_POST['submit'])){
   $dirname = $_POST['dirname'];
   if (is_dir($dirname)) {
      echo "<h2>Contenuto della directory $dirname:</h2>";
      $dir = opendir($dirname);
      while($file = readdir($dir)) {
         echo "$file<br />";
      }
      closedir($dir);
   }
   else
      echo "$dirname non è una direcyory.";
}

?>
<html>
   <body bgcolor="#FFFFFF" text="#000000">
   <h2>Contenuto della directory</h2>

   <form name="form1" method="post" action="<?php echo
     $_SERVER['PHP_SELF']; ?> ">

   Nome della directory:<input type="text" name="dirname">
   <input type="submit" name="submit" value="Invia nome file">
   <input type="reset" name="Reset" value="Reimposta">

   </form>

   </body>
</html>
```

Lo script è una nuova versione degli esempi proposti in precedenza in questo capitolo. La parte più interessante è la sezione in cui il nome della directory viene verificato per stabilire se si tratta effettivamente di una directory, quindi il contenuto della directory viene letto e visualizzato con un ciclo `while`.

```
$dirname = $_POST['dirname'];
if (is_dir($dirname)) {
   echo "<h2>Contenuto della directory $dirname:</h2>";
   $dir = opendir($dirname);
   while($file = readdir($dir)) {
      echo "$file<br />";
   }
   closedir($dir);
```

L'output ottenuto da questo script è quello della Figura 27.4.

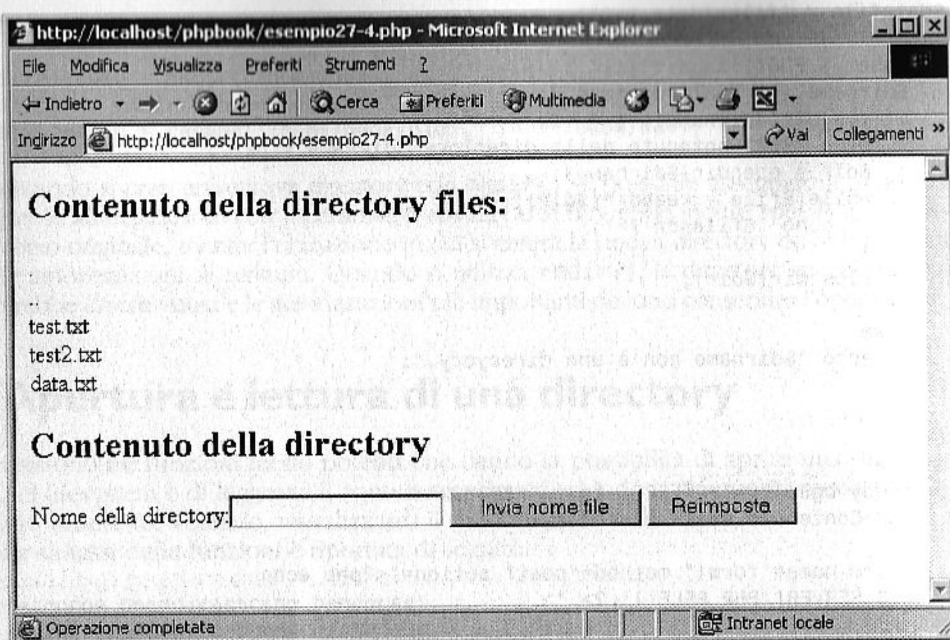


Figura 27.4
Visualizzazione del contenuto della directory.

Riepilogo

In questo capitolo sono state introdotte altre funzioni utilizzabili per modificare e creare file e directory. Avete visto come utilizzare le funzioni per ottenere statistiche relative a un file, cancellare e spostare file e directory e persino visualizzare l'intero contenuto di una directory. Nel prossimo capitolo sarà introdotto il concetto di caricamento di file.

Capitolo 28

Caricamento di file tramite i moduli

Introduzione

I moduli consentono agli utenti di inserire dati tramite campi, come i campi per il testo o per le password; i moduli sono stati introdotti nel Capitolo 18. In quella trattazione è stato omesso un tipo di campo che consente a un utente di inviare un intero file assieme al modulo. Il tipo di input "file" per il moduli consente agli utenti di sfogliare le directory del computer locale e selezionare un file da caricare sul server Web.
Una volta caricato, è possibile scrivere uno script PHP per accedere a questo file, spostarlo, archiviarlo e, come si vedrà, utilizzarlo a vantaggio dell'utente.

Tipo di input file per i moduli

Per dar vita a un modulo che consenta di caricare un file, è necessario inserire l'attributo `enctype` nell'elemento `form` con il valore `multipart/form-data`, in modo da informare il server che il modulo potrebbe contenere un file caricato:

```
<form enctype='multipart/form-data' action='form.php' method='post'>
```

All'elemento `input` è possibile assegnare l'attributo `type="file"` per creare un campo di input del modulo che consenta di sfogliare le directory della macchina locale e selezionare un file da caricare sul server. Il formato di questo elemento è il seguente:

```
<input name='myFileName' type='file'>
```

È inoltre possibile limitare la dimensione massima in byte attraverso un campo nascosto in cui la variabile di sistema `MAX_FILE_SIZE` è impostata su un certo valore. Di seguito se ne riporta un esempio:

```
<input type='hidden' name='MAX_FILE_SIZE' value='100000'>
```

Quello che segue è invece un esempio di modulo contenente un campo per il caricamento di file.

```
<?php

// Caricamento di file - Esempio 28-1
//---------------------

?>

<form enctype='multipart/form-data' action='<?php echo
$_SERVER["PHP_SELF"];
?>' method='post'>

<input type='hidden' name='MAX_FILE_SIZE' value='100000'>
File di documento: <input name='myfile' type='file'
value='$file'><br><br>
<input type='submit' value='Invia il documento' name='send'>
</form>
```

La Figura 28.1 illustra come appare questo modulo quando viene visualizzato in un browser.

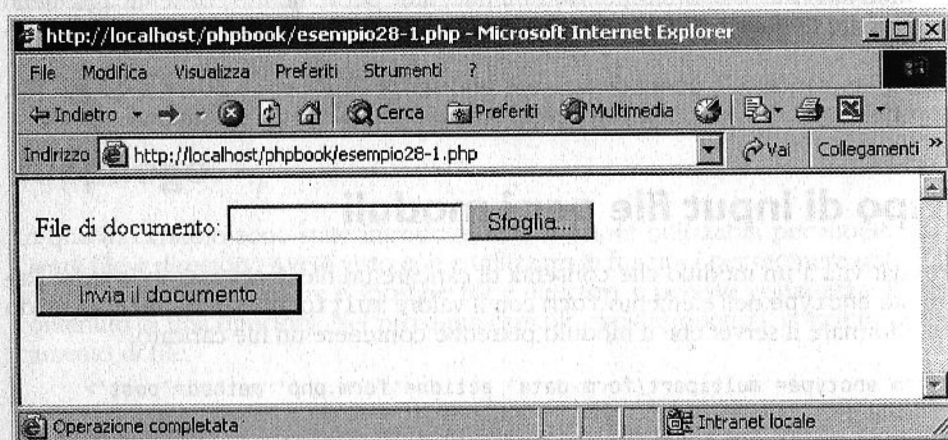


Figura 28.1
Un semplice modulo per il caricamento di file.

Potete inserire il nome e il percorso del file da caricare nel campo del modulo oppure fare clic sul pulsante *Sfoglia* per aprire una finestra di selezione dei file che consenta di individuare il file da caricare. Questa finestra è quella della Figura 28.2. Sfortunatamente, anche se il modulo è stato creato, non esiste ancora un codice PHP in grado di accedere a esso.

Accesso a un file caricato con PHP

I passi da compiere per scrivere uno script PHP in grado di accedere a un file caricato sono diversi. È necessario muoversi con prudenza, in modo da agire corretta-

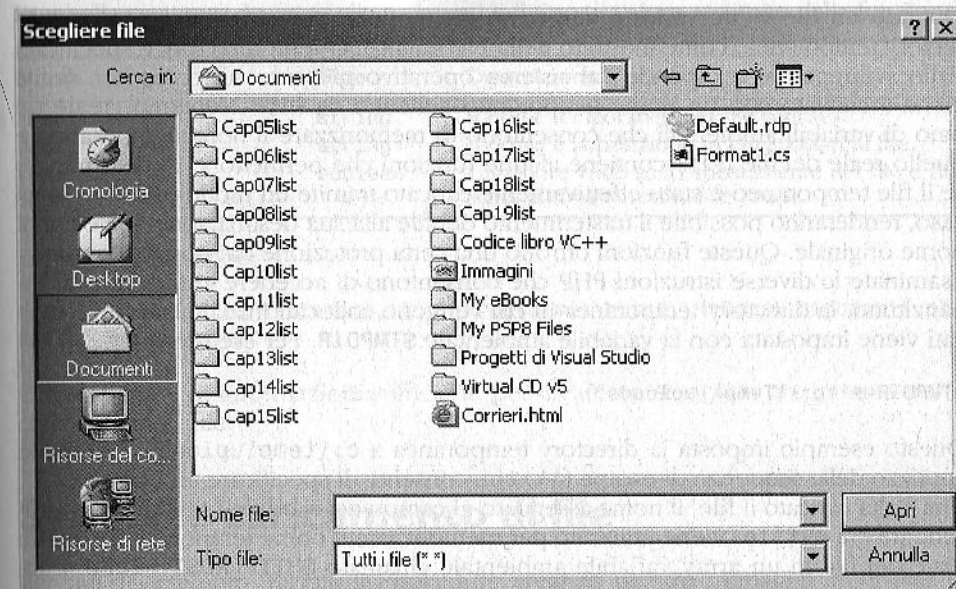


Figura 28.2
Finestra per la selezione di un file.

mente, in quanto si tratta di un'operazione che comporta alcuni rischi per la sicurezza. Prima di affrontare la procedura da seguire, osservate come funziona il caricamento di file in PHP. Il processo è illustrato nella Figura 28.3.

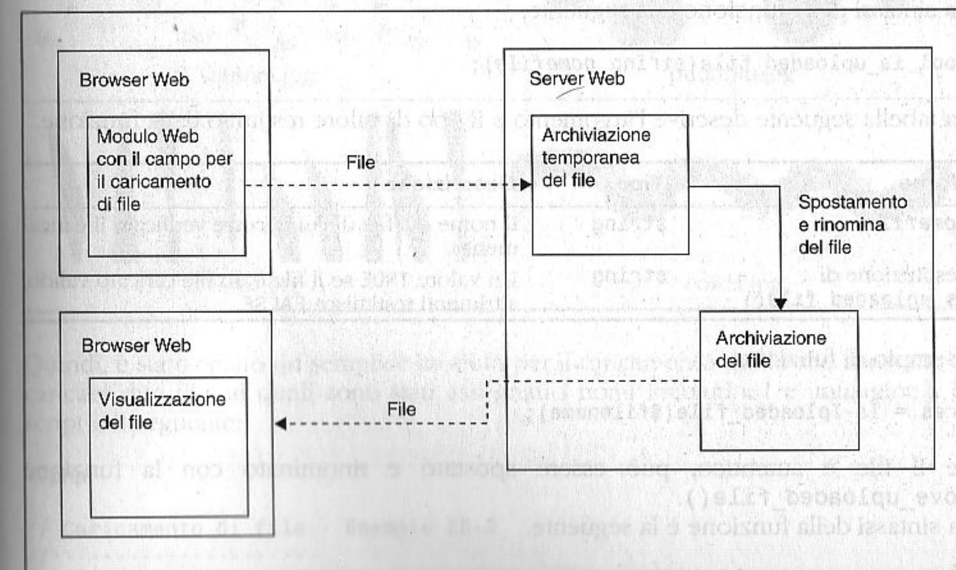


Figura 28.3
Funzionamento del caricamento di file in PHP.

Quando un file viene caricato attraverso un modulo, viene archiviato in una directory temporanea, come mostrato nella Figura 28.3. Questa directory è specificata dallo sviluppatore e dipende dal sistema operativo presente sul computer. Il file viene archiviato con un nome temporaneo generato da PHP. Vengono create un paio di variabili ambientali che consentono di memorizzare il nome temporaneo e quello reale del file. PHP contiene alcune funzioni che permettono di determinare se il file temporaneo è stato effettivamente caricato tramite un modulo PHP e, in tal caso, renderanno possibile il trasferimento del file alla sua destinazione finale con il nome originale. Queste funzioni offrono una certa protezione dai possibili intrusi. Esaminate le diverse istruzioni PHP che consentono di accedere ai file caricati. Innanzitutto, la directory temporanea in cui vengono collocati inizialmente i file caricati viene impostata con la variabile ambientale `$TMPDIR`. Per esempio:

```
$TMPDIR = "c:\\Temp\\uploads";
```

Questo esempio imposta la directory temporanea a `c:\temp\uploads`. Osservate l'utilizzo della sequenza di escape (`\\`) che consente di specificare un carattere "`\`". Una volta caricato il file, il nome assegnato al campo del modulo (nell'esempio precedente era `myfile`) viene utilizzato per memorizzare il nome temporaneo e quello reale del file in un array variabile ambientale chiamato `$HTTP_POST_FILES`. L'accesso al nome temporaneo e a quello reale del file può avvenire nel modo seguente:

```
$filename = $HTTP_POST_FILES['myfile']['tmp_name'];
$realname =$HTTP_POST_FILES['myfile']['name'];
```

È importante accedere a queste informazioni, in quanto dovete utilizzarle per accertare che si tratti di un file autentico caricato correttamente. A tal fine utilizzate la funzione `is_uploaded_file()`.
La sintassi della funzione è la seguente.

```
bool is_uploaded_file(string nomefile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
`nomefile`	`string`	Il nome del file di cui occorre verificare il caricamento.
Restituzione di `is_uploaded_file()`	`string`	Un valore `TRUE` se il file è un file caricato valido, altrimenti restituisce `FALSE`.

Esempio di funzione:

```
$res = Is-Iploaded_file($filename);
```

Se il file è autentico, può essere spostato e rinominato con la funzione `move_uploaded_file()`.
La sintassi della funzione è la seguente.

```
bool move_uploaded_file(string nomefile, string destinazione);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
`nomefile`	`string`	Il nome temporaneo del file caricato.
`destinazione`	`string`	Il nome e la posizione in cui spostare il file.
Restituzione di `move_uploaded_file()`	`boolean`	Un valore `TRUE` se lo spostamento del file è riuscito, altrimenti restituisce `FALSE`.

Esempio di funzione:

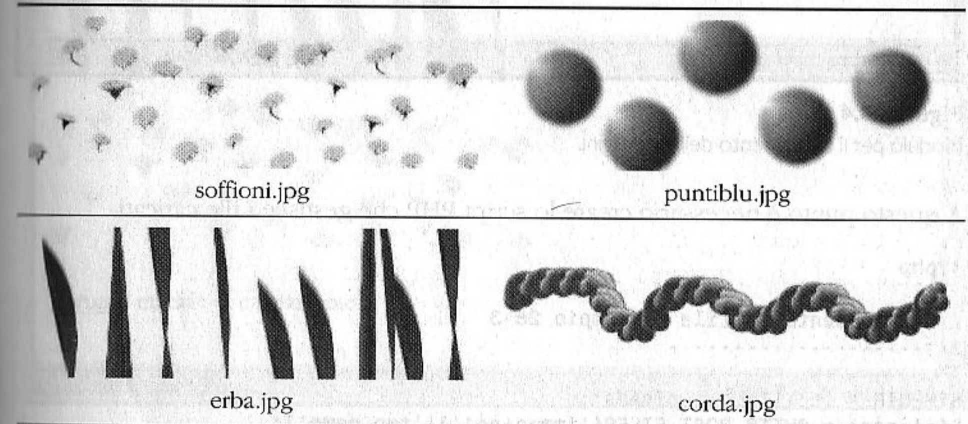
```
move_uploaded_file($filename,("c:\\Programmi\\httpd\\HtDocs\\bigbook\\uploads\\".$newname));
```

Ora disponete di informazioni sufficienti per creare un semplice programma per il caricamento di file.

Semplice caricamento di file

Per questo esempio iniziate la procedura creando quattro immagini archiviate in *c:\temp\pics*. Queste immagini sono state salvate come corda.jpg, puntiblu.jpg, soffioni.jpg ed erba.jpg e sono quelle della Tabella 28.1.

Tabella 28.1 I file delle immagini con i loro nomi



Quindi, è stato creato un semplice modulo per il caricamento di file che consente di caricare due file, ai quali sono stati assegnati i nomi immagine1 e immagine2; lo script è il seguente:

```
<?php

// Caricamento di file - Esempio 28-2
//---------------------

?>

<form enctype='multipart/form-data' action='esempio28-3.php 'method='post'>
```

```
<input type='hidden' name='MAX_FILE_SIZE' value='1000000'>
Prima immagine: <input name='immagine1' type='file'><br>
Seconda immagine: <input name='immagine2' type='file'><br><br>
<input type='submit' value='Invia le immagini' name='send'>

</form>
```

Questo modulo è quello della Figura 28.4. Il modulo è stato progettato in modo da caricare due delle quattro immagini create in precedenza, benché non vi sia alcun controllo di errore che garantisca che i file caricati siano effettivamente immagini.

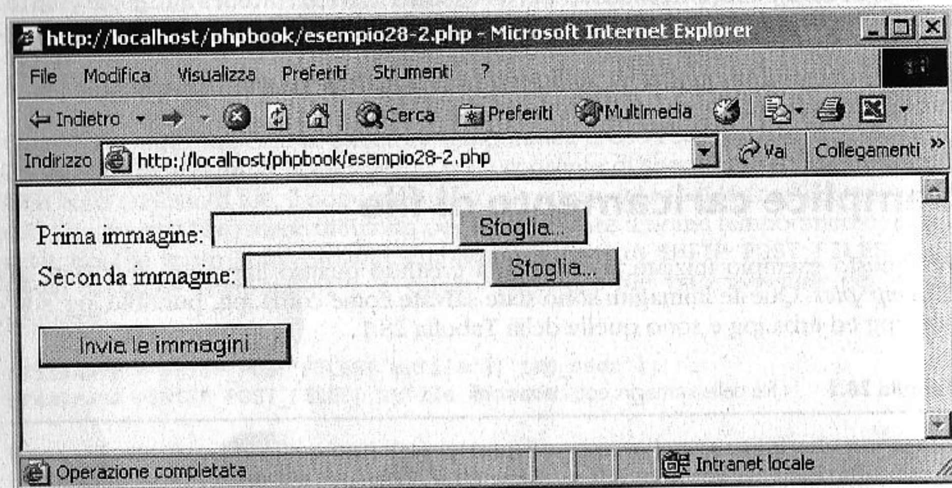


Figura 28.4
Modulo per il caricamento delle immagini.

A questo punto è necessario creare lo script PHP che gestisce i file caricati.

```
<?php

// Caricamento di file - Esempio 28-3
//----------------------

$TMPDIR = "c:\\Temp\\uploads";
$filename = $HTTP_POST_FILES['immagine1']['tmp_name'];
if (is_uploaded_file($filename))
   move_uploaded_file($filename, "c:\\Inetpub\\wwwroot\\phpbook\\
                      graphics\\immagine1.jpg");
$filename = $HTTP_POST_FILES['immegine2']['tmp_name'];
if (is_uploaded_file($filename))
   move_uploaded_file($filename, "c:\\Inetpub\\wwwroot\\phpbook\\
                      graphics\\immagine2.jpg");

?>
<img src="graphics\immagine1.jpg">
<br>
<img src="graphics\immagine2.jpg">
<br>
<br>
<a href="esempio28-2.php">Torna</a> al modulo di caricamento.
```

Lo script verifica che il primo file (immagine1) sia un file valido e lo sposta in:

```
"c:\\Inetpub\\wwwroot\\phpbook\\graphics\\immagine1.jpg"
```

Poi è il secondo file (immagine2) a essere verificato e spostato in:

```
"c:\\Inetpub\\wwwroot\\phpbook\\graphics\\immagine2.jpg"
```

Infine, lo script visualizza le due immagini con l'elemento `<IMG SRC>` e fornisce un collegamento ipertestuale per tornare al modulo di caricamento. Un esempio di output di questo script è quello della Figura 28.5. Ovviamente le immagini visualizzate dipenderanno dalle immagini che sceglieste di caricare.

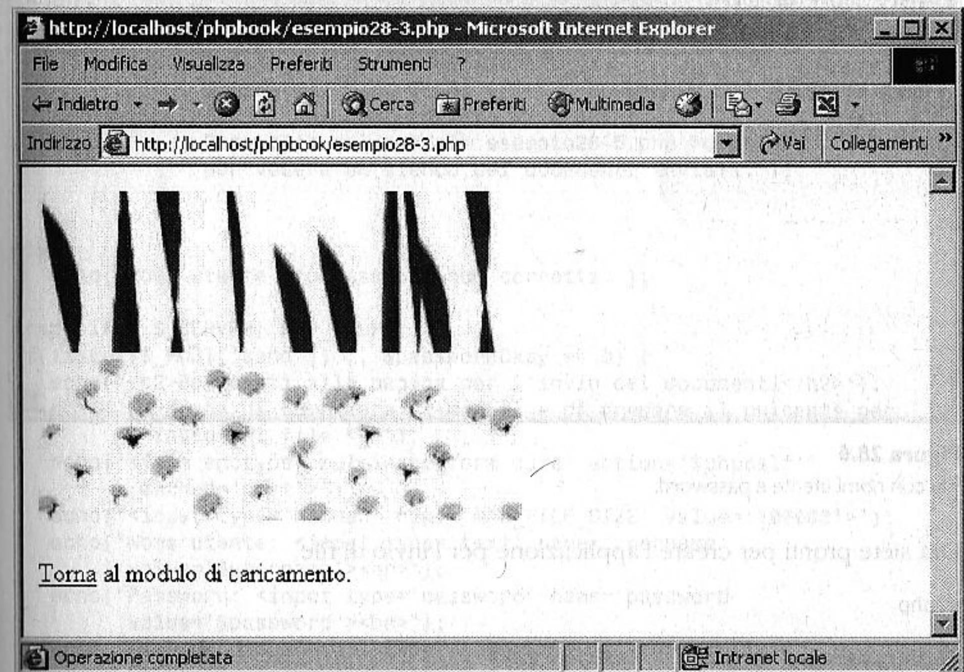


Figura 28.5
Visualizzazione delle immagini caricate.

Protezione del caricamento di file

Benché l'esempio precedente descriva il funzionamento del meccanismo di caricamento e manipolazione dei file, esso mette anche in luce un importante problema: la sicurezza. Nell'esempio precedente non viene effettuato alcun controllo su chi carica le immagini, pertanto chiunque conosca l'indirizzo della pagine del modulo di caricamento può caricare e sostituire quelle installate in precedenza. Per ora que-

sto non comporta gravi conseguenze, ma se il modulo fosse concepito per caricare file relativi a documenti che solo determinati utenti hanno l'autorizzazione a inviare, sarebbe necessario un ambiente più sicuro.

Per ovviare al problema, occorre innanzitutto creare un meccanismo che consenta l'identificazione di chi sta cercando di caricare un documento e di stabilire se si desidera consentire l'operazione. Un semplice sistema con nome utente e password potrebbe essere sufficiente. A tal fine, per prima cosa si utilizza Blocco note per creare un semplice file dove ogni coppia di nome utente e password è elencata su righe separate del file. Ogni nome utente è separato dalla propria password da uno spazio. La Figura 28.6 illustra un esempio di file di questo genere, che può essere salvato con il nome password.txt. Non salvatelo nella directory del server Web, altrimenti i potenziali hacker potranno accedervi e visualizzare l'elenco dei nomi utente con le relative password.

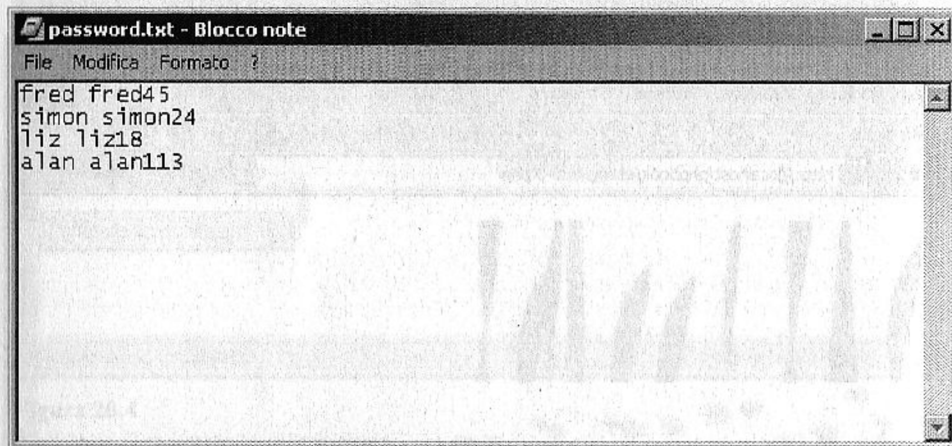


Figura 28.6

File con nomi utente e password.

Ora siete pronti per creare l'applicazione per l'invio di file.

```
<?php

// Caricamento di file - Esempio 28-4
//---------------------

if (isset($_POST['username']))
  $username = $_POST['username'];
else
  $username = "";
if (isset($_POST['password']))
  $password = $_POST['password'];
else
  $password = "";
if (isset($_POST['file']))
  $file = $_POST['file'];
else
  $file = "";

$passwordOkay = 0;

if (isset($_POST['send'])) {
   $okay = checkUsernamePassword("c:\\Temp\\password.txt",
          $username,$password);
   if($okay) {
      $passwordOkay = 1;
      $TMPDIR = "c:\\Temp\\uploads";
      $filename = $HTTP_POST_FILES['myfile']['tmp_name'];
      $realname = $HTTP_POST_FILES['myfile']['name'];
      if (is_uploaded_file($filename)) {
         $date = fDate();
         $time = fTime();
         $newname = ($realname . "-" . $date . "-" . $time);
         move_uploaded_file($filename,("c:\\Inetpub\\wwwroot\\
                     phpbook\\uploads\\".$newname));
         appendToFile("uploads\\uploads.txt",
                  $date . " ". $time . " " . $username . " " . $realname);
         echo("Grazie per aver inviato il documento.
               Fare clic su <a href='esempio28-5.php'>qui</a>
               per vedere un elenco dei documenti inviati.");
      }
   }
else
   echo("Nome utente e/o password non corretti.");
}
$phpself = $_SERVER['PHP_SELF'];
if(!isset($_POST['send']) || $passwordOkay == 0) {
   echo("<h2>Benvenuti alla pagina per l'invio dei documenti</h2>");
   echo("Si prega di compilare il modulo e di premere il pulsante per
         l'invio del file.<P>");
   echo("<form enctype='multipart/form-data' action='$phpself'
          method='post'>");
   echo("<input type='hidden' name='MAX_FILE_SIZE' value='100000'>");
   echo("Nome utente: <input type='text' name='username'
         value='$username'><br>");
   echo("Password: <input type='password' name='password'
         value='$password'><br>");
   echo("File di documento: <input name='myfile' type='file'
         value='$file'><br><br>");
   echo("<input type='submit' value='Invia il documento' name='send'>");
   echo("</form>");
}

function fDate() {
   $date = getdate();
   $monthText = $date["month"];
   $year = $date["year"];
   $mday = $date["mday"];
   return $mday . "-" . $monthText . "-" . $year;
}

function fTime() {
   $time = localtime();
```

```
    return $time[2] . "-" . $time[1] . "-" . $time[0];
}

function appendToFile($file,$data) {
    $out = fopen($file,"a");
    fputs($out,$data."\n");
    fclose($out);
}

function checkUsernamePassword($file,$username,$password) {
    $found=0;
    $in = fopen($file,"r");
    $line = fgets($in,4096);
    while(!feof($in) && !$found) {
        $splitLine = explode (" ", $line);
        $splitLine[1] = substr($splitLine[1],0,strlen($splitLine[1])-2);
        if($splitLine[0] == $username && $splitLine[1] == $password)
            $found=1;
        $line = fgets($in,4096);
    }
    fclose($in);
    return $found;
}

?>
```

È un po' più lungo rispetto al primo esempio, ma non preoccupatevi: sarà esaminato in dettaglio per spiegare cosa accade nelle varie fasi. La prima parte dello script ottiene le variabili del modulo:

```
if (isset($_POST['username']))
  $username = $_POST['username'];
else
  $username = "";
if (isset($_POST['password']))
  $password = $_POST['password'];
else
  $password = "";
if (isset($_POST['file']))
  $file = $_POST['file'];
else
  $file = "";

$passwordOkay = 0;
```

La parte successiva dello script verifica che i dati del modulo siano stati inviati e, in tal caso, richiama la funzione `checkUsernamePassword()`, passandole come parametri la posizione del file delle password, il nome utente e la password:

```
if (isset($_POST['send'])) {
  $okay = checkUsernamePassword("c:\\Temp\\password.txt",
          $username,$password);
```

La funzione `checkUsernamePassword()` restituisce 0 se il nome utente o la password non esistono oppure 1 se va tutto bene. Si tornerà in seguito su questa funzione quando ne verrà fornita la definizione, più avanti nel listato. Se la password e il nome utente sono validi, la parte successiva dello script imposta la directory temporanea e accede ai nomi temporanei e reali del file caricato; quindi verifica che si tratti di un file caricato valido chiamando la funzione `is_uploaded_file()`.

```
  if($okay) {
    $passwordOkay = 1;
    $TMPDIR = "c:\\Temp\\uploads";
    $filename = $HTTP_POST_FILES['myfile']['tmp_name'];
    $realname = $HTTP_POST_FILES['myfile']['name'];
    if (is_uploaded_file($filename)) {
```

Se il file deriva da un caricamento valido, la data e l'ora vengono ottenute chiamando le funzioni `fDate()` e `fTime()`. La data e l'ora restituite da queste funzioni vengono utilizzate per creare un nuovo nome di file che incorpora il nome di file reale assieme alla data e ora:

```
      $date = fDate();
      $time = fTime();
      $newname = ($realname . "-" . $date . "-" . $time);
```

Il file viene quindi spostato in una directory precedentemente creata, chiamata *uploads*. Se volete, potete cambiare la posizione di questa directory:

```
      move_uploaded_file($filename,("c:\\Inetpub\\wwwroot\\
                    phpbook\\uploads\\".$newname));
```

Viene chiamata la funzione `appendToFile()`, cui vengono passati come parametro il nome del file da scrivere, la data, l'ora, il nome utente e il nome reale del file come singola stringa. Infine, viene visualizzato un messaggio che conferma l'invio del documento e chiede all'utente se desidera visualizzare un elenco dei documenti inviati:

```
      appendToFile("uploads\\uploads.txt",
            $date . " ". $time . " " . $username . " " . $realname);
      echo("Grazie per aver inviato il documento.
            Fare clic su <a href='esempio28-5.php'>qui</a>
            per vedere un elenco dei documenti inviati.");
    }
```

Se il nome utente e la password non sono corretti, viene visualizzato un messaggio che informa l'utente dell'errore:

```
else
    echo("Nome utente e/o password non corretti.");
```

Se il modulo non è stato inviato, o il nome utente/password non sono corretti, viene visualizzato il modulo, costituito da campi che consentono all'utente di inserire il nome utente e la password e di selezionare il documento da inviare:

```
$phpself = $_SERVER['PHP_SELF'];
if(!isset($_POST['send']) || $passwordOkay == 0) {
  echo("<h2>Benvenuti alla pagina per l'invio dei documenti</h2>");
  echo("Si prega di compilare il modulo e di premere il pulsante per
        l'invio del file.<P>");
```

```
    echo("<form enctype='multipart/form-data' action='$phpself'
          method='post'>");
    echo("<input type='hidden' name='MAX_FILE_SIZE' value='100000'>");
    echo("Nome utente: <input type='text' name='username'
          value='$username'><br>");
    echo("Password: <input type='password' name='password'
          value='$password'><br>");
    echo("File di documento: <input name='myfile' type='file'
          value='$file'><br><br>");
    echo("<input type='submit' value='Invia il documento' name='send'>");
    echo("</form>");
}
```

L'ulteriore parte del programma è costituita dalle funzioni definite dall'utente chiamate in precedenza; le prime due sono `fDate()` e `fTime()`, che restituiscono la data e l'ora come stringhe:

```
function fDate() {
   $date = getdate();
   $monthText = $date["month"];
   $year = $date["year"];
   $mday = $date["mday"];
   return $mday . "-" . $monthText . "-" . $year;
}

function fTime() {
   $time = localtime();
   return $time[2] . "-" . $time[1] . "-" . $time[0];
}
```

La funzione `appendToFile()` scrive un record relativo al documento appena caricato nel file uploads.txt. Questo file di testo viene utilizzato dal secondo programma PHP per visualizzare i documenti che sono stati caricati:

```
function appendToFile($file,$data) {
   $out = fopen($file,"a");
   fputs($out,$data."\n");
   fclose($out);
}
```

La parte finale del programma è costituita dalla funzione `checkUsernamePassword()`, che apre il file passwords.txt e ne legge tutte le righe:

```
function checkUsernamePassword($file,$username,$password) {
   $found=0;
   $in = fopen($file,"r");
   $line = fgets($in,4096);
   while(!feof($in) && !$found) {
```

Ogni riga del file viene separata con la funzione `explode()` per ottenere il nome utente e la password memorizzata:

```
      $splitLine = explode (" ", $line);
```

Dato che i file di testo spesso inseriscono caratteri di avanzamento riga alla fine di ogni riga, questi vengono rimossi con la funzione `substr()`, che elimina gli ultimi due caratteri nascosti dalla stringa:

```
      $splitLine[1] = substr($splitLine[1],0,strlen($splitLine[1])-2);
```

Il nome utente e la password passati alla funzione vengono confrontati con quelli letti dal file e, se corrispondono, la variabile `$found` viene impostata a 1; questo valore viene quindi restituito alla funzione:

```
      if($splitLine[0] == $username && $splitLine[1] == $password)
         $found=1;
      $line = fgets($in,4096);
   }
   fclose($in);
   return $found;
}
```

L'output prodotto da questo programma è quello della Figura 28.7.

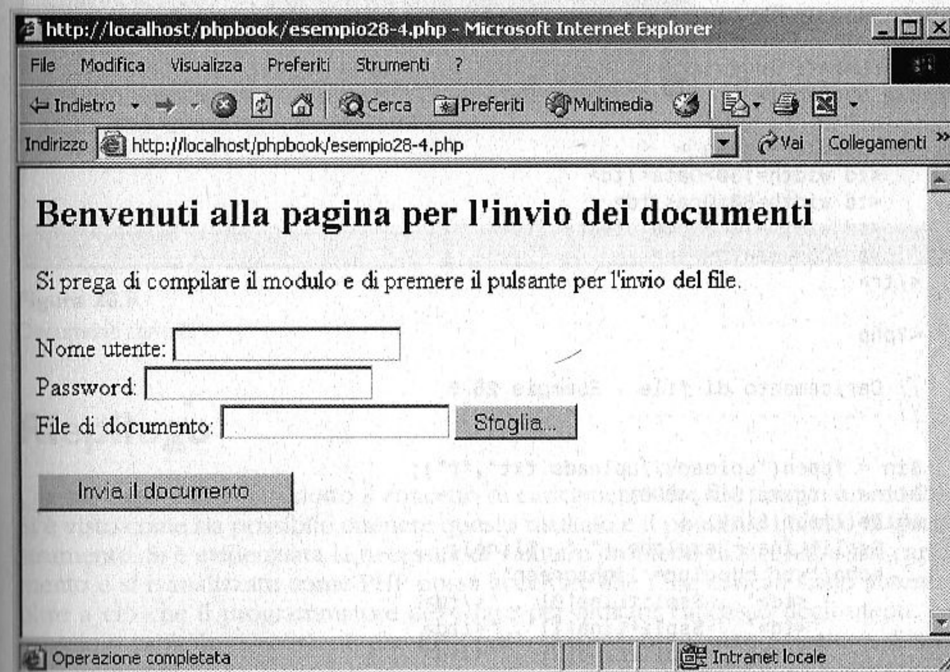


Figura 28.7
La schermata per il caricamento di file sicuri.

Se vengono inseriti un nome utente e una password corretti e quindi viene caricato un file, appare il messaggio della Figura 28.8.

Lo stadio successivo consiste nella creazione del programma che visualizza i documenti caricati quando l'utente seleziona il collegamento ipertestuale *qui*. Il pro-

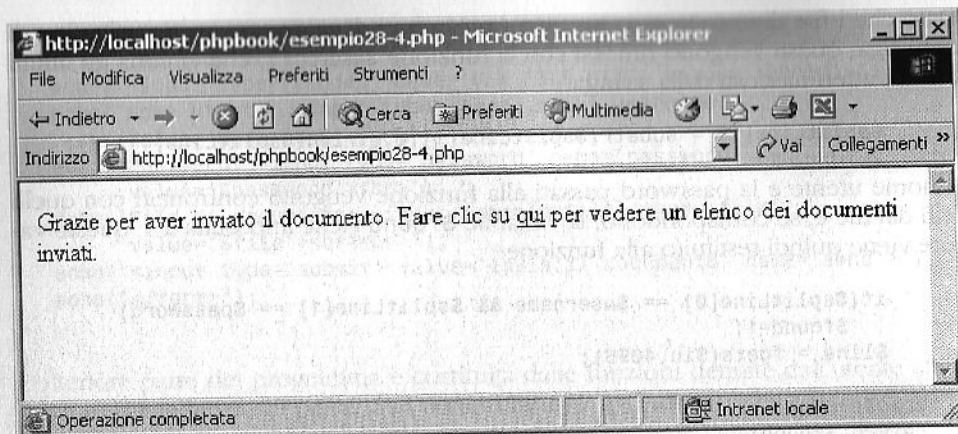


Figura 28.8
Caricamento riuscito.

gramma si limita ad aprire il file uploads.txt e a leggerne tutte le righe: I dettagli relativi a ciascun file caricato vengono visualizzati in una tabella:

```
<h2>Documenti inviati</h2>
<table border=1>

   <tr bgcolor='cyan'>
      <td width=130>Data</td>
      <td width=80>Ora</td>
      <td width=100>Nome utente</td>
      <td>Documento</td>
   </tr>

   <?php

   // Caricamento di file - Esempio 28-5
   //---------------------

   $in = fopen("uploads//uploads.txt","r");
   $line = fgets($in,4096);
   while(!feof($in)) {
      $splitLine = explode (" ", $line);
      echo("<tr bgcolor='lightgreen'>
            <td>" . $splitLine[0] . "</td>
            <td>" . $splitLine[1] . "</td>
            <td>" . $splitLine[2] . "</td>
            <td>
            <a href='uploads\\" . $splitLine[3] . "-" . $splitLine[0] .
             "-" . $splitLine[1] . "'>" . $splitLine[3] . "</a>
            </td>
            </tr>");
      $line = fgets($in,4096);
   }
   fclose($in);

   ?>
```



```
</table>
<a href='esempio28-4.php'>Torna</a> alla pagina di invio dei documenti.
```

Un esempio di output di questo programma è riportato nella Figura 28.9.
L'utente può accedere ai file caricati facendo clic sul nome del documento. In relazione al tipo di documento, esso sarà visualizzato (come nel caso di un file di immagine) oppure, nel caso si tratti di un documento, il sistema potrebbe chiedere all'utente se desidera scaricare una copia del file sul proprio computer.

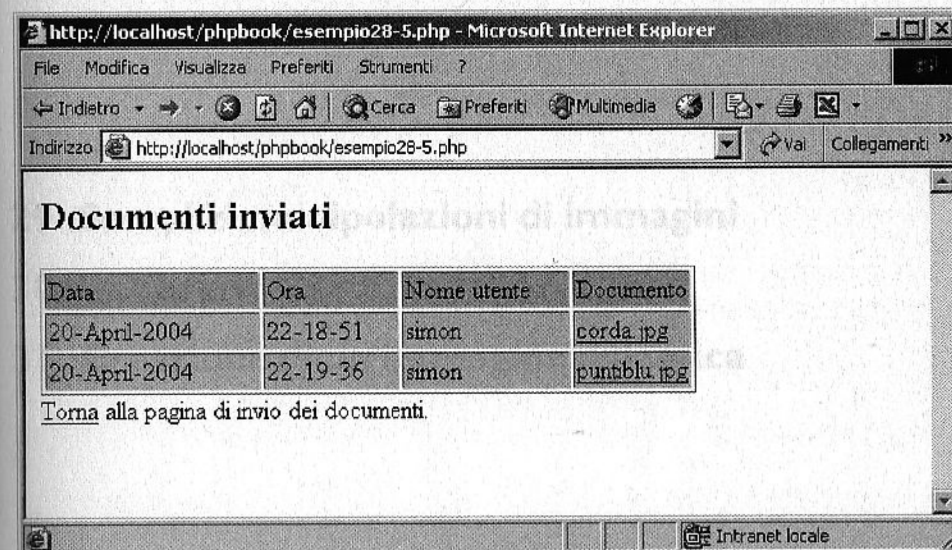


Figura 28.9
Documenti caricati.

Riepilogo

Questo capitolo ha introdotto il concetto di caricamento dei file tramite un modulo. Si è visto come sia possibile ottenere questo risultato e il possibile utilizzo di questo strumento. Si è evidenziata la necessità di valutare problemi di sicurezza nel caricamento e si è analizzato come PHP possa accertare che i file caricati siano autentici, oltre a ciò che il programmatore deve fare per limitare l'accesso degli utenti. Nel prossimo capitolo si vedrà ciò che PHP può ottenere con la manipolazione di semplici immagini.

Capitolo 29
Semplici manipolazioni di immagini

Introduzione

Questo capitolo illustra come utilizzare PHP per eseguire semplici manipolazioni di immagini. La capacità di eseguire questo genere di modifiche consentirà di sviluppare siti Web più dinamici, dall'aspetto molto ricercato e con una migliore accessibilità. Le manipolazioni di immagini che analizzeremo sono operazioni semplici, come scegliere quale immagine visualizzare oppure ridimensionare e ingrandire un'immagine. Malgrado la loro semplicità, queste operazioni permettono di ottenere risultati di tutto rispetto.

Sostituzione di un'immagine

Una tra le funzioni più semplici che PHP è in grado di gestire è il controllo delle immagini che compaiono sui siti Web. Molto spesso i progettisti Web desiderano che i propri siti appaiano come se fossero in costante evoluzione, senza richiedere un'ec-



Figura 29.1
Una normalissima immagine Web.

cessiva manutenzione. Molti progettisti, per esempio, decidono di inserire nei propri siti una serie di immagini che cambiano in base al periodo dell'anno.
Prendete in considerazione per un attimo la Figura 29.1 che illustra un tipico logo Web di un'azienda.
L'obiettivo è fare in modo che il logo cambi in base alla data in cui viene visualizzata la pagina Web. Quindi, se l'utente visualizza la pagina la vigilia di Natale, vedrà il logo rappresentato nella Figura 29.2, mentre il giorno di Natale comparirà il logo illustrato nella Figura 29.3.



Figura 29.2
Logo per la vigilia di Natale.

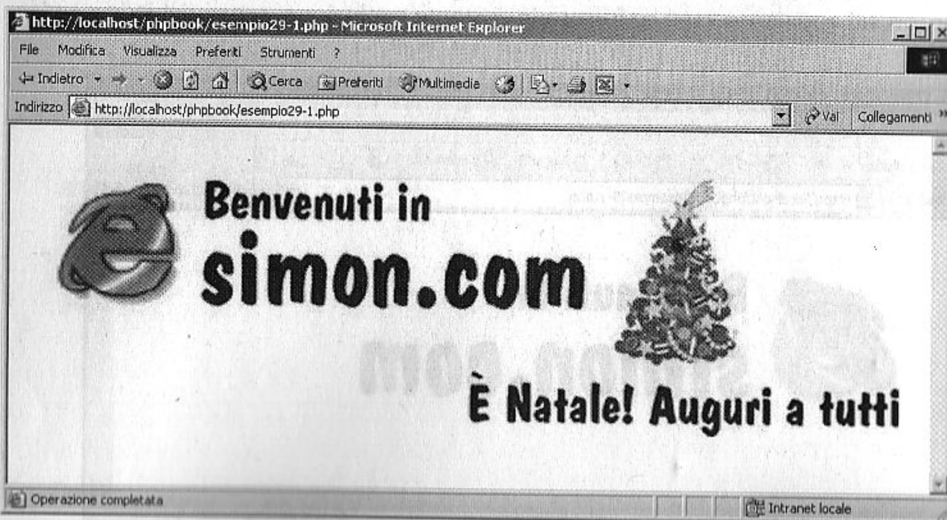


Figura 29.3
Logo per il giorno di Natale.

Per fortuna, il codice necessario per apportare queste modifiche è molto semplice.

```
<?php

// Semplici immagini - Esempio 29-1
//---------------------------------

$date = getdate();
$day = $date["mday"];
$month = $date["mon"];
if($day == 25 && $month== 12)
   echo("<img src='graphics/simoncomna.jpg'>");
elseif($day == 24 && $month== 12)
   echo("<img src='graphics/simoncomvn.jpg'>");
else
   echo("<img src='graphics/simoncom.jpg'>");

?>
```

Affinché questo programma funzioni occorrono tre immagini separate tra cui scegliere. Le immagini a nostra disposizione sono: simoncomna.jpg, simoncomvn.jpg e simoncom.jpg. La Tabella 29.1 mostra queste tre immagini e i relativi nomi di file.

Tabella 29.1 Immagini dello script

Welcome to simon.com	Simoncom.jpg
Welcome to simon.com It's Christmas Eve	Simoncomce.jpg
Welcome to simon.com It's Christmas Day	Simoncomcd.jpg

Il programma inizia ottenendo la data di sistema, quindi memorizza il giorno e il mese in due variabili: `$day` e `$month`. Per questo esempio l'anno non è necessario:

```
$date = getdate();
$day = $date["mday"];
$month = $date["mon"];
```

Quindi, utilizza un costrutto **if**, **elseif** – **else** per controllare le date in modo da decidere quale immagine visualizzare:

```
if($day == 25 && $month== 12)
   echo("<img src='graphics/simoncomna.jpg'>");
```

```
elseif($day == 24 && $month== 12)
   echo("<img src='graphics/simoncomvn.jpg'>");
else
   echo("<img src='graphics/simoncom.jpg'>");
```

Quindi, il giorno di Natale comparirà l'immagine `simoncomna.jpg`, la vigilia di Natale comparirà l'immagine 'simoncomvn.jpg', mentre tutti gli altri giorni dell'anno l'utente vedrà il logo simoncom.jpg. Se volete provare le modifiche apportate senza aspettare Natale o la vigilia, inserite le due righe di codice seguenti prima dell'istruzione `if` per impostare il mese e il giorno:

```
$month = 12;
$day = 24;
```

Ridimensionamento delle immagini

Le immagine utilizzate nelle pagine Web hanno una dimensione fissa, in termini di pixel. In genere queste immagini sono visualizzate nella dimensione predefinita con l'elemento `<img src="nomeimmagine">`. Questo elemento ha due attributi, altezza e larghezza, che consentono di specificare la dimensione esatta con cui visualizzare l'immagine, a prescindere dalla dimensione fisica dell'immagine stessa.
Esiste la possibilità di alterare in modo dinamico i valori di altezza e larghezza di un'immagine, facendola apparire con nuove dimensioni. La qualità dell'immagine visualizzata dipende dalla dimensione originale dell'immagine stessa. Per esempio, un'immagine di 10 × 10 pixel ingrandita a 100 × 100 pixel avrà un aspetto "quadrettato", poiché ogni pixel originale è stato moltiplicato per dieci.

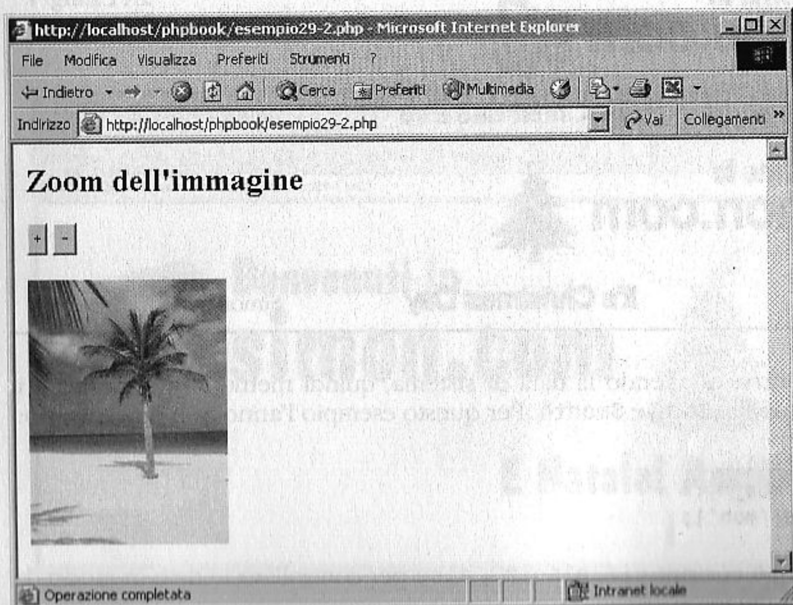


Figura 29.4
Esempio di ridimensionamento di un'immagine.

La Figura 29.4 mostra l'esempio di un'immagine ridimensionata. La pagina Web è costituita dall'immagine di una palma e da un semplice modulo con due pulsanti, uno contrassegnato con un segno "+", l'altro con un segno "–". Un clic sul pulsante "+" ingrandisce l'immagine, mentre un clic sul pulsante "–" la rimpicciolisce.
La Figura 29.5 mostra la medesima immagine dopo che è stata leggermente rimpicciolita con un clic sul pulsante "–". Nel programma sono stati incorporati alcuni controlli per limitare la dimensione minima e massima di ingrandimento o riduzione dell'immagine.

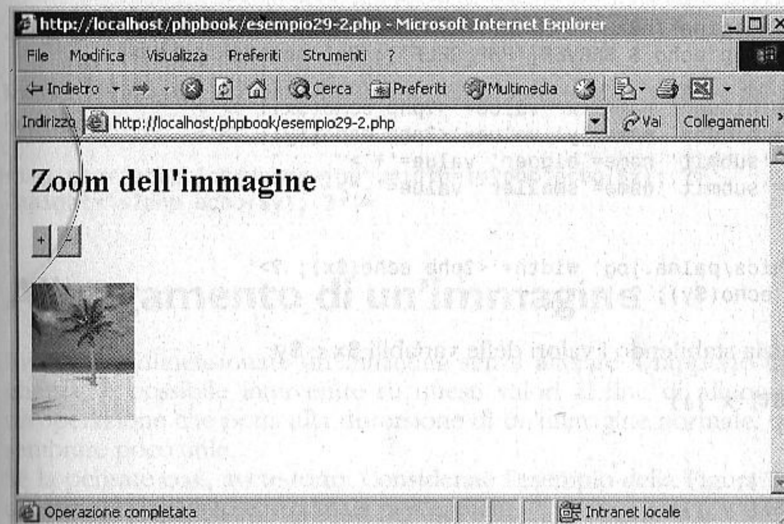


Figura 29.5
Un'immagine rimpicciolita.

Il codice necessario per implementare questa pagina Web è il seguente:

```
<?php

// Semplici immagini - Esempio 29-2
//------------------------------

if(!isset($_POST['x']))
   $x=150;
else
   $x=$_POST['x'];
if(!isset($_POST['y']))
   $y=200;
else
   $y=$_POST['y'];
if (isset($_POST['bigger'])) {
   if($_POST['bigger']) {
      if($x < 300){
         $x=$x*2;
         $y=$y*2;
      }
   }
```

```
}
if (isset($_POST['smaller'])) {
   if($_POST['smaller']) {
      if($x > 38){
         $x=$x/2;
         $y=$y/2;
      }
   }
}

?>
<h2>Zoom dell'immagine</h2>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method='post'>

   <input type='hidden' name='x' value='<?php echo($x); ?>'>
   <input type='hidden' name='y' value='<?php echo($y); ?>'>
   <input type='submit' name='bigger' value='+'>
   <input type='submit' name='smaller' value='-'>

</form>
<img src='graphics/palma.jpg' width='<?php echo($x); ?>'
 height='<?php echo($y); ?>'>
```

Il programma inizia stabilendo i valori delle variabili $x e $y:

```
if(!isset($_POST['x']))
   $x=150;
else
   $x=$_POST['x'];
if(!isset($_POST['y']))
   $y=200;
else
   $y=$_POST['y'];
```

Successivamente, se l'utente preme il pulsante "+" i valori di $x e $y vengono raddoppiati, purché il valore di $x sia minore di 300:

```
if (isset($_POST['bigger'])) {
   if($_POST['bigger']) {
      if($x < 300){
         $x=$x*2;
         $y=$y*2;
      }
   }
}
```

Se invece l'utente preme il pulsante "–" i valori di $x and $y vengono dimezzati, purché il valore di $x sia maggiore di 38:

```
if (isset($_POST['smaller'])) {
   if($_POST['smaller']) {
      if($x > 38){
         $x=$x/2;
         $y=$y/2;
      }
   }
}
```


La sezione successiva del codice visualizza il modulo Web che include i valori nascosti delle variabili $x e $y:

```
<h2>Zoom dell'immagine</h2>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method='post'>

   <input type='hidden' name='x' value='<?php echo($x); ?>'>
   <input type='hidden' name='y' value='<?php echo($y); ?>'>
   <input type='submit' name='bigger' value='+'>
   <input type='submit' name='smaller' value='-'>

</form>
```

Infine, l'immagine viene visualizzata utilizzando i valori di $x e $y per impostarne la dimensione:

```
<img src='graphics/palma.jpg' width='<?php echo($x); ?>'
 height='<?php echo($y); ?>'>
```

Allungamento di un'immagine

Invece di ridimensionare un'immagine senza alterare il rapporto tra altezza e larghezza, è possibile intervenire su questi valori al fine di allungarla. Si tratta di un'operazione che porta alla distorsione di un'immagine normale, quindi potrebbe sembrare poco utile.
Se la pensate così, avete torto. Considerate l'esempio della Figura 29.6, che illustra un semplice modulo e una scala percentuale.

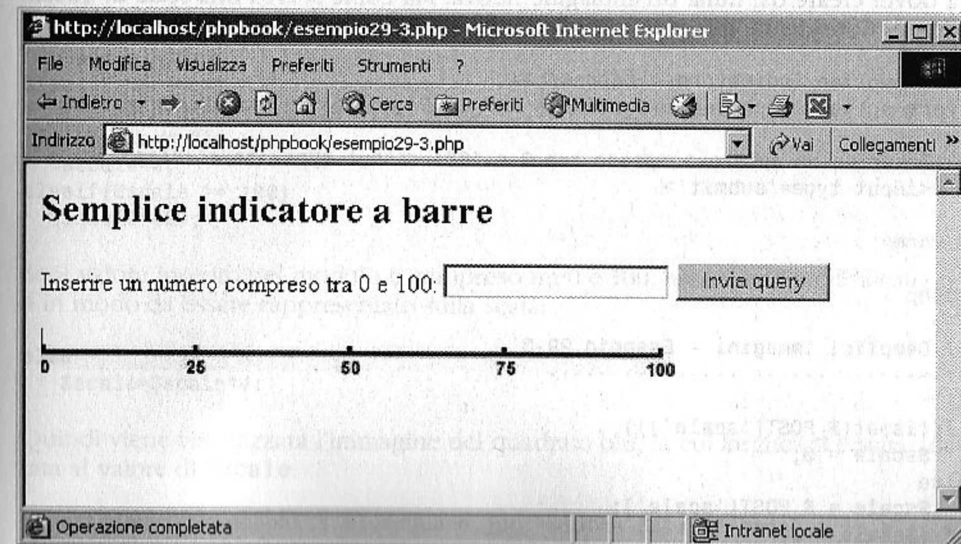


Figura 29.6
Un semplice indicatore a barre.

Se l'utente inserisce nel modulo un valore compreso tra 0 e 100 e poi fa clic sul pulsante di invio, il valore compare sulla barra di scala, come mostrato nella Figura 29.7.

Figura 29.7
Barra impostata al 67%.

La capacità di creare barre di scala come questa è molto utile, poiché esse mettono a disposizione uno strumento grafico per rappresentare i dati nella pagina Web senza dover creare dal nulla un'immagine nuova. Ma come si crea una scala di questo genere? Considerate il seguente codice sorgente:

```
<h2>Semplice indicatore a barre</h2>
<form action='<?php echo($_SERVER['PHP_SELF']); ?>' method='post'>

  Inserire un numero compreso tra 0 e 100: <input type='text' name='scale'>
  <input type='submit'>

</form>

<?php

// Semplici immagini - Esempio 29-3
//-------------------------------

if(!isset($_POST['scale']))
  $scale = 0;
else
  $scale = $_POST['scale'];
if ($scale < 0)
  $scale=0;
elseif($scale >= 100)
  $scale=400;
else
  $scale=$scale*4;
echo("<img src='graphics/bluesquare.jpg' width='$scale' height='10'>");

?>
<br><img src='graphics/barscale.jpg'>
```


Il programma inizia richiedendo la creazione di due immagini. La prima è barscale.jpg, una semplice scala numerica. La seconda immagine è un quadrato blu 10 × 10 pixel. Queste immagini sono rappresentate nella Tabella 29.2.

Tabella 29.2 Immagini della barra

25 50 75 100	Barscale.jpg
	Bluesquare.jpg

Quest'ultima immagine è quella che sarà utilizzata per creare la barra sulla scala. Da un'analisi dello script, si può notare che la prima sezione visualizza il modulo in cui l'utente inserisce il valore da visualizzare:

```
<h2>Semplice indicatore a barre</h2>
<form action='<?php echo($_SERVER['PHP_SELF']); ?>' method='post'>
Inserire un numero compreso tra 0 e 100: <input type='text' name='scale'>
<input type='submit'>
</form>
```

La sezione successiva controlla che i valori inseriti siano compresi tra 0 e 100 e, in caso negativo, li imposta a 0 o 400; poiché la scala è larga 400 pixel, infatti, la barra deve essere larga 400 pixel per rappresentare il 100%:

```
if(!isset($_POST['scale']))
  $scale = 0;
else
  $scale = $_POST['scale'];
if ($scale < 0)
  $scale=0;
elseif($scale >= 100)
  $scale=400;
```

Se il valore inserito nel modulo è compreso tra 0 e 100, esso viene moltiplicato per 4 in modo da essere rappresentato sulla scala:

```
else
  $scale=$scale*4;
```

Quindi viene visualizzata l'immagine del quadrato blu, la cui larghezza è stata adattata al valore di **$scale**:

```
echo("<img src='graphics/bluesquare.jpg' width='$scale' height='10'>");
```

Infine, compare l'immagine della barra:

```
<br><img src='graphics/barscale.jpg'>
```

Barre di scala simili sono molto impiegate nelle pagine Web per rappresentare in veste grafica i dati ottenuti da database o file. La natura dinamica dei grafici permette all'immagine di fornire un quadro aggiornato dei dati.

Riepilogo

Questo capitolo ha mostrato ciò che PHP è in grado di ottenere manipolando immagini già esistenti.
Anche se gli esempi proposti sono abbastanza banali, in realtà possono essere di grande effetto. Il prossimo capitolo approfondirà l'argomento della manipolazione di immagini e presenterà la libreria GD di PHP, che consente di creare immagini da zero.

Capitolo 30

La libreria GD

Introduzione

La maggior parte delle pagine Web contiene immagini utilizzate per migliorare l'aspetto e la fruizione delle pagine stesse. Nel capitolo precedente si è visto che PHP può essere utilizzato per manipolare immagini esistenti; tuttavia i progettisti Web devono prestare attenzione nell'utilizzo delle immagini, poiché un abuso di grafica può infastidire gli utenti e ridurre pertanto l'accessibilità. PHP ha una libreria di funzioni che consente di creare e manipolare immagini. Con PHP, creazione e visualizzazione delle immagini possono avvenire utilizzando un numero molto ridotto di righe di codice. In questo capitolo verranno introdotte alcune librerie di funzioni disponibili, tuttavia dovete ricordare ciò che si è detto sulla prudenza nel ricorso alle immagini. Il semplice fatto che PHP offra strumenti per creare e manipolare le immagini non significa che dobbiate abusarne.

Come procurarsi la libreria GD

Le funzioni per la manipolazione delle immagini di PHP sono fornite dalla libreria GD, che ora viene fornita come standard con l'installazione di PHP per MS Windows (supponendo che abbiate scaricato il pacchetto zip binario e non il file di installazione più piccolo). Dovrete tuttavia modificare il file php.ini per attivarla. Questo file si trova nella directory *c:\windows* o *c:\winnt* di un sistema Windows. Per attivare la libreria GD dovete aprire il file php.ini in un editor come Blocco note e poi trovare la riga seguente nel file:

```
;extension=php_gd2.dll
```

Per attivare la libreria è sufficiente rimuovere il ; dall'inizio della riga e salvare il file. La Figura 30.1 illustra il file php.ini visualizzato in Blocco note. Dovrete anche accertarvi che la directory *extensions* punti al percorso del file gd2.dll sul vostro computer. Nell'esempio di questo libro è il seguente:

```
extension_dir = "c:\php\extensions\"
```

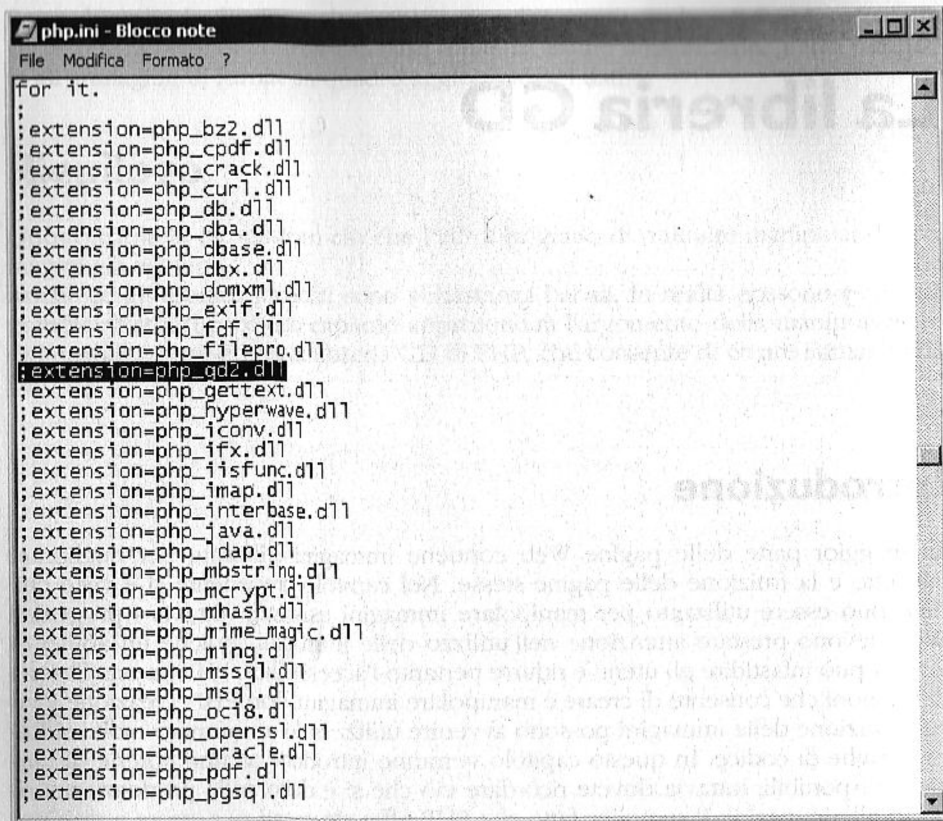


Figura 30.1
Il file php.ini.

Versioni della libreria GD

Le prime versioni della libreria GD offrivano il supporto per la creazione di immagini nei formati JPEG (*Joint Photographics Experts Group*) e GIF (*Graphic Interchange Format*). Tuttavia, a causa della controversia legale relativa alla proprietà del formato GIF, le versioni più recenti della libreria GD hanno rimosso il supporto per la creazione di immagini di questo tipo, introducendo al suo posto il supporto per le immagini PNG (*Portable Network Graphics*), un formato di file progettato per la grafica Web. Questo capitolo si concentrerà sulla creazione di immagini JPEG e PNG.

Creazione di una semplice immagine JPEG

Si comincerà illustrando come creare un'immagine JPEG molto semplice. A tal fine è necessario introdurre diverse funzioni della libreria GD, la prima delle quali è `ImageCreate()`.

La sintassi della funzione è la seguente.

```
int ImageCreate(int x, int y);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
x	int	Larghezza dell'immagine in pixel.
y	int	Altezza dell'immagine in pixel.
Restituzione di `imageCreate()`	int	La funzione `ImageCreate()` restituisce un valore intero che viene utilizzato per memorizzare l'handle dell'immagine.

Esempio di funzione:

```
$image = ImageCreate(100,100);
```

Questa istruzione creerà un'immagine di dimensioni di 100 × 100 pixel. La funzione successiva è `ImageColorAllocate()`.
La sintassi della funzione è la seguente.

```
int ImageColorAllocate(int immagine, int rosso, int verde, int blu);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
rosso	int	Valore che specifica la quantità di rosso utilizzata per formare il colore. I valori sono compresi tra 0 e 255.
verde	int	Valore che specifica la quantità di verde utilizzata per formare il colore. I valori sono compresi tra 0 e 255.
blu	int	Valore che specifica la quantità di blu utilizzata per formare il colore. I valori sono compresi tra 0 e 255.
Restituzione di `imageColorAllocate()`	int	Restituisce un identificatore di colore al colore specificato.

Esempio di funzione:

```
$red = ImageColorAllocate($image,255,0,0);
```

Questo esempio crea il colore rosso, che viene memorizzato nella variabile `$red`. Poi è possibile utilizzare la funzione `ImageFill()` per "riempire" l'immagine con un colore particolare.
La sintassi della funzione è la seguente.

```
int ImageFill(int immagine, int x, int y, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione ImageCreate().
x	int	La coordinata X per l'inizio del riempimento.
y	int	La coordinata Y per l'inizio del riempimento.
colore	int	Il colore da utilizzare. Questo valore viene ricevuto dalla funzione ImageColorAllocate.
Restituzione di imageFill()	int	Restituisce 1 (TRUE) se va tutto bene o 0 (FALSE) se si verifica un errore.

Esempio di funzione:

```
ImageFill($image,0,0,$red);
```

Questo esempio riempirà l'immagine **$image** a partire dal suo angolo superiore sinistro con il colore specificato con la variabile **$red**.
A questo punto siete pronti per creare e memorizzare l'immagine. A tal fine utilizzerete la funzione ImageJPEG().
La sintassi della funzione è la seguente.

```
Int ImageJPEG(int immagine, string nomefile);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione imageCreate.
nomefile	string	Il nome del file dell'immagine.
Restituzione di imageJPEG()	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageJPEG($image,'quadratorosso.jpeg');
```

Questo esempio creerà un'immagine chiamata quadratorosso.jpeg. Infine viene utilizzata la funzione ImageDestroy() per liberare la memoria sfruttata nella generazione dell'immagine.
La sintassi della funzione è la seguente.

```
int ImageDestroy(int immagine);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione ImageCreate().
Restituzione di imageDestroy()	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageDestroy($image);
```

Lo script che segue utilizza tutte queste funzioni insieme, per produrre una semplice immagine.

```
<?php

// Immagini GD - Esempio 30-1
//---------------------

$image = ImageCreate(100,100);
$red =ImageColorAllocate($image,255,0,0);
ImageFill ($image,0,0,$red);
ImageJPEG($image, 'graphics/quadratorosso.jpeg');
ImageDestroy($image);

?>
<img src='graphics/quadratorosso.jpeg'>
```

L'output di questo script produce qualcosa di simile a ciò che è illustrato nella Figura 30.2.

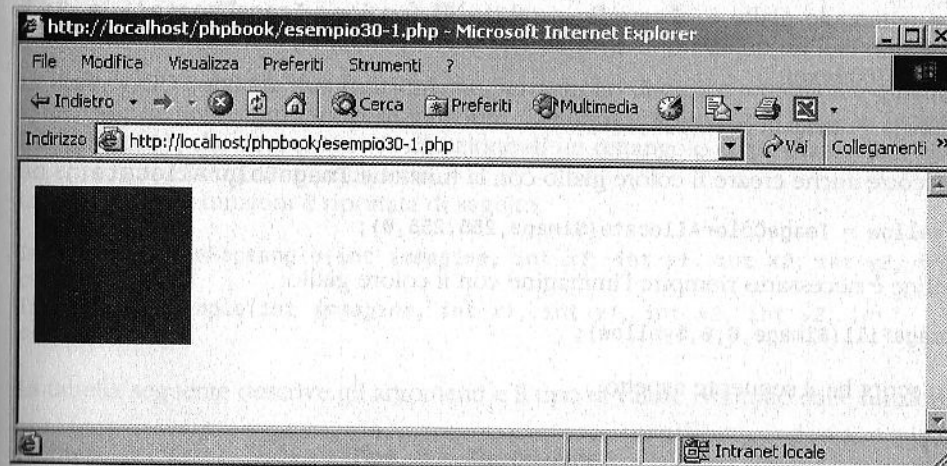


Figura 30.2
Una semplice immagine JPEG.

Se controllate la directory del server Web che contiene i file .php, vedrete che contiene un'immagine chiamata quadratorosso.jpeg. Questa immagine viene visualizzata sulla pagina Web grazie all'elemento HTML `<img src="quadratorosso.jpeg">`. L'immagine generata può essere visualizzata e modificata con un qualunque programma di manipolazione grafica.

Creazione di una semplice immagine PNG

Il programma PHP precedente dà vita a un'immagine JPEG. Qualora vogliate creare un'immagine PNG, il programma sarebbe molto simile: cambia soltanto una riga di codice, in quanto è necessario utilizzare la funzione `ImagePNG()`.
La sintassi della funzione è la seguente.

```
Int ImagePNG(int immagine, string nomefile);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate ()`.
nomefile	string	Il nome del file di immagine.
Restituzione di `imagePNG()`	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImagePNG($image,'rettangologiallo.png');
```

Questa istruzione creerà un'immagine di nome rettangologiallo.png. Per dar vita a un rettangolo giallo è necessario cambiare la funzione `ImageCreate()` in modo che non crei un quadrato. L'esempio che segue genera un'immagine di dimensioni 200 × 100 pixel:

```
$image = ImageCreate(200,100);
```

Occorre anche creare il colore giallo con la funzione `ImageColorAllocate()`:

```
$yellow = ImageColorAllocate($image,255,255,0);
```

Infine è necessario riempire l'immagine con il colore giallo:

```
ImageFill($image,0,0,$yellow);
```

Lo script ha il seguente aspetto:

```
<?php

// Immagini GD - Esempio 30-2
//--------------------

$image = ImageCreate(200,100);
$yellow = ImageColorAllocate($image,255,255,0);
ImageFill($image,0,0,$yellow);
ImagePNG($image, 'graphics/rettangologiallo.png');
ImageDestroy($image);

?>
<img src='graphics/rettangologiallo.png'>
```

La visualizzazione dell'output di questo script in un browser dovrebbe produrre qualcosa di simile a quanto illustrato nella Figura 30.3.

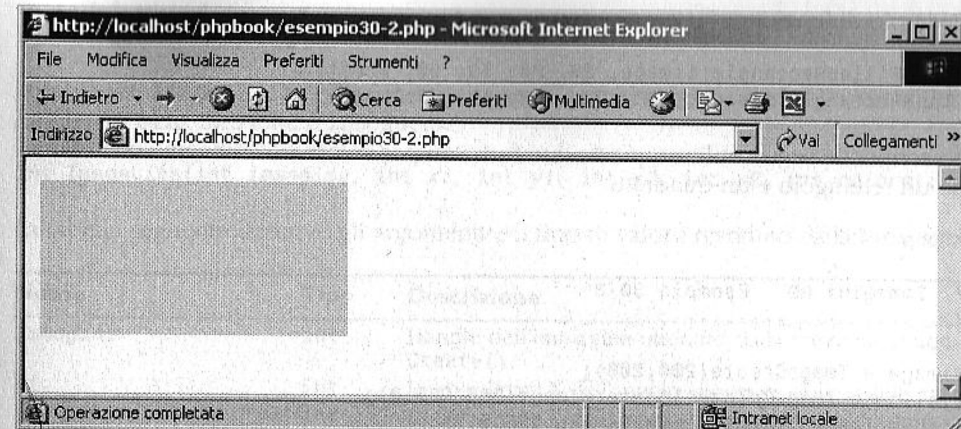


Figura 30.3
Una semplice immagine PNG.

Un esame della directory Web contenente le immagini confermerà la presenza di un file di nome rettangologiallo.png.

Disegno di rettangoli e quadrati

La libreria GD contiene due funzioni che permettono di creare rettangoli e quadrati sulle proprie immagini: sono le funzioni `ImageFilledRectangle()` e `ImageRectangle()`, che consentono la creazione di un rettangolo o quadrato pieno e di un contorno rettangolare o quadrato.
La sintassi delle funzioni è riportata di seguito.

```
Int ImageFilledRectangle(int immagine, int x1, int y1, int x2, int y2, int colore);
Int ImageRectangle(int immagine, int x1, int y1, int x2, int y2, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
x1	int	La coordinata X dell'angolo superiore sinistro del rettangolo.
y1	int	La coordinata Y dell'angolo superiore sinistro del rettangolo.
x2	int	La coordinata X dell'angolo inferiore destro del rettangolo.
y2	int	La coordinata Y dell'angolo inferiore destro del rettangolo.
colore	int	Colore del rettangolo.
Restituzione di `imageFilledRectangle()` e `imageRectangle()`	int	Restituiscono 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageFilledRectangle($image, 20, 20, 50, 50, $red);
ImageRectangle($image, 100, 20, 150, 150, $blue);
```

Lo script che segue utilizza queste due funzioni per creare un'immagine che contiene un rettangolo e un quadrato.

```
<?php

// Immagini GD - Esempio 30-3
//---------------------

$image = ImageCreate(200,200);
$yellow = ImageColorAllocate($image,255,255,0);
$red = ImageColorAllocate($image,255,0,0);
$blue = ImageColorAllocate($image,0,0,255);
ImageFill($image,0,0,$yellow);
ImageFilledRectangle($image, 20, 20, 50, 50, $red);
ImageRectangle($image, 100, 20, 150, 150, $blue);
ImagePNG($image, 'graphics/rettangoli.png');
ImageDestroy($image);

?>
<img src='graphics/rettangoli.png'>
```

La visualizzazione dell'output di questo script dovrebbe produrre qualcosa di simile a quanto mostrato nella Figura 30.4.

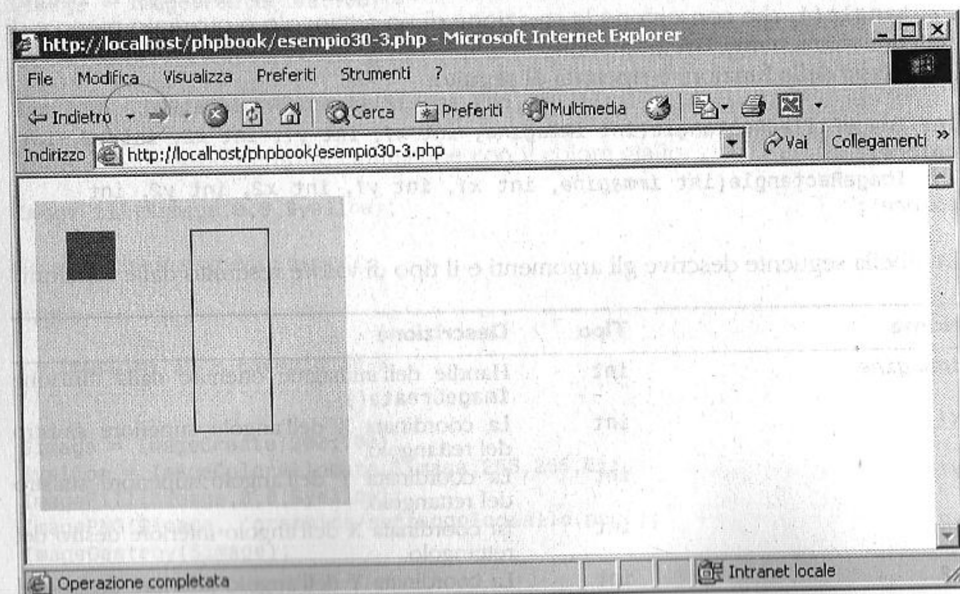


Figura 30.4
Rettangoli e quadrati.

Disegnare righe continue

la funzione `ImageLine()` offre un mezzo per disegnare righe dritte sulle immagini. La sintassi della funzione è la seguente.

```
Int ImageLine(int immagine, int x1, int y1, int x2, int y2, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
x1	int	La coordinata X dell'angolo superiore sinistro della riga.
y1	int	La coordinata Y dell'angolo superiore sinistro della riga.
x2	int	La coordinata X dell'angolo inferiore destro della riga.
y2	int	La coordinata Y dell'angolo inferiore destro della riga.
colore	int	Colore delle riga.
Restituzione di `imageLine()`	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageLine($image, 10,10,50,50,$red);
```

Considerate lo script seguente:

```
<?php

// Immagini GD - Esempio 30-4
//---------------------

$image = imagecreate(101,101);
$yellow = ImageColorAllocate($image,255,255,0);
$black = ImageColorAllocate($image,0,0,0);
ImageFill($image,0,0,$yellow);
for($x=0;$x<=100;$x+=10) {
   ImageLine($image,$x,0,$x,100,$black);
}
for($y=0;$y<=100;$y+=10) {
   ImageLine($image,0,$y,100,$y,$black);
}
ImagePNG($image, 'graphics/grigliagialla.png');
ImageDestroy($image);

?>
<img src='graphics/grigliagialla.png'>
```

Lo script illustra un esempio di utilizzo della funzione `ImageLine()`. Esaminandolo in modo più dettagliato si nota che viene creata un'immagine di 101 × 101 pixel e vengono definiti due colori, giallo e nero:

```
$image = imagecreate(101,101);
$yellow = ImageColorAllocate($image,255,255,0);
$black = ImageColorAllocate($image,0,0,0);
```

Poi l'immagine viene riempita con il colore giallo:

```
ImageFill($image,0,0,$yellow);
```

Le righe di codice successive utilizzano un ciclo `for` per creare 11 righe che appaiono dall'alto verso il basso dell'immagine, staccate di 10 pixel:

```
for($x=0;$x<=100;$x+=10) {
  ImageLine($image,$x,0,$x,100,$black);
}
```

Le righe di codice successive creano 11 righe da sinistra a destra dell'immagine, staccate di 10 pixel:

```
for($y=0;$y<=100;$y+=10) {
  ImageLine($image,0,$y,100,$y,$black);
}
```

Infine l'immagine viene salvata, la memoria utilizzata per la sua creazione viene rilasciata e l'immagine viene visualizzata:

```
ImagePNG($image, 'graphics/grigliagialla.png');
ImageDestroy($image);

?>
<img src='graphics/grigliagialla.png'>
```

L'output prodotto da questo programma è quello della Figura 30.5.

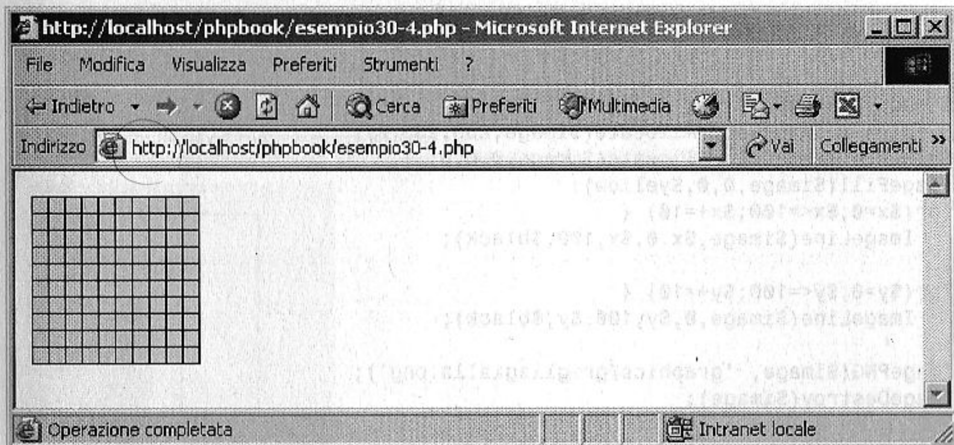


Figura 30.5
Disegnare righe continue.

Disegno di righe tratteggiate

La funzione `ImageSetStyle()` viene utilizzata in combinazione con la funzione `ImageLine()` per descrivere il formato della riga.

La sintassi della funzione è la seguente.

```
Int ImageSetStyle(int immagine, array stile);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
stile	array	Array di pixel che descrivono la riga.
Restituzione di `imageSetStyle()`	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageSetStyle ($image, $style);
```

Lo script che segue offre un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-5
//--------------------

$image = imagecreate(100,100);
$lightblue = ImageColorAllocate($image,64,64,255);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$lightblue);

$style = array ($white, $lightblue);

ImageSetStyle($image,$style);
ImageLine($image,10,50,90,50,IMG_COLOR_STYLED);
ImageLine($image,50,10,50,90,IMG_COLOR_STYLED);
ImagePNG($image, 'graphics/tratteggioblu.png');
ImageDestroy($image);

?>
<img src='graphics/tratteggioblu.png'>
```

Osservate che `$style` è definita come semplice array di punti alternati bianchi e blu:

```
$style = array ($white, $lightblue);
```

Questo viene poi definito come stile:

```
ImageSetStyle($image,$style);
```

Infine viene chiamata la funzione `ImageLine()`, che è in grado di utilizzare lo stile definito attraverso la costante `IMG_COLOR_STYLED`:

```
ImageLine($image,10,50,90,50,IMG_COLOR_STYLED);
ImageLine($image,50,10,50,90,IMG_COLOR_STYLED);
```

La Figura 30.6 illustra l'output prodotto da questo script.

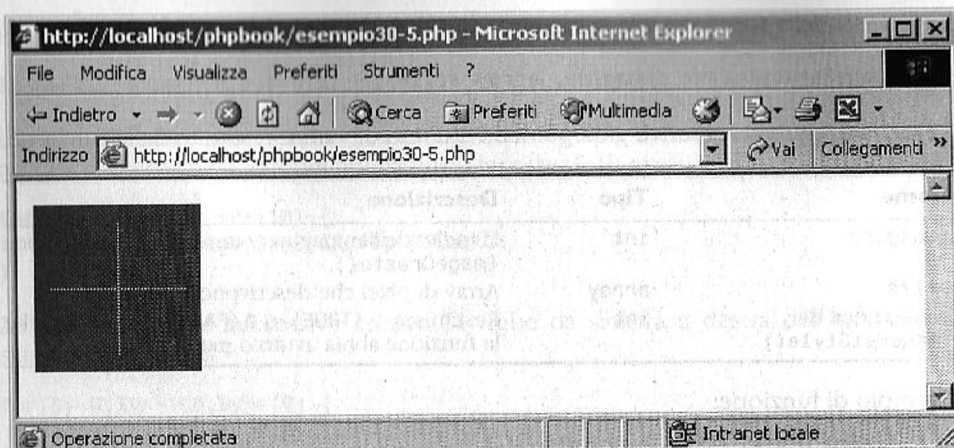


Figura 30.6
Disegnare righe tratteggiate.

Inserimento di testo

La libreria GD supporta funzioni che consentono di inserire testo in un'immagine. Si tratta della funzione `ImageString()`, che permette l'inserimento di testo da sinistra a destra, e della funzione `ImageStringUp()`, che consente di inserire testo dall'alto in basso.
La sintassi delle funzioni è riportata di seguito.

```
Int ImageString(int immagine, int dimensione, int x, int y, string testo,
int colore);
Int ImageStringUp(int immagine, int dimensione, int x, int y, string
testo, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
dimensione	int	La dimensione del testo da visualizzare.
x	int	La coordinata X dell'angolo superiore sinistro del testo.
y	int	La coordinata Y dell'angolo superiore sinistro del testo.
testo	string	Il testo da visualizzare.
colore	int	Colore del rettangolo.
Restituzione di `imageString()` e `imageStringUp()`	int	Restituisce 1 (TRUE) o 0 (FALSE) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageString($image,4,10,10,"Ciao", $white);
```


Lo script seguente illustra un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-6
//---------------------

$image = imagecreate(100,100);
$darkred = ImageColorAllocate($image,192,0,0);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$darkred);
ImageString($image,4,10,10,"Ciao,", $white);
ImageString($image,3,10,25,"io", $white);
ImageString($image,2,10,40,"mi", $white);
ImageString($image,1,10,55,"chiamo", $white);
ImageString($image,5,10,65,"Simon", $white);
ImagePNG($image, 'graphics/testorosso.png');
ImageDestroy($image);

?>
<img src='graphics/testorosso.png'>
```

La Figura 30.7 illustra l'output prodotto da questo script.

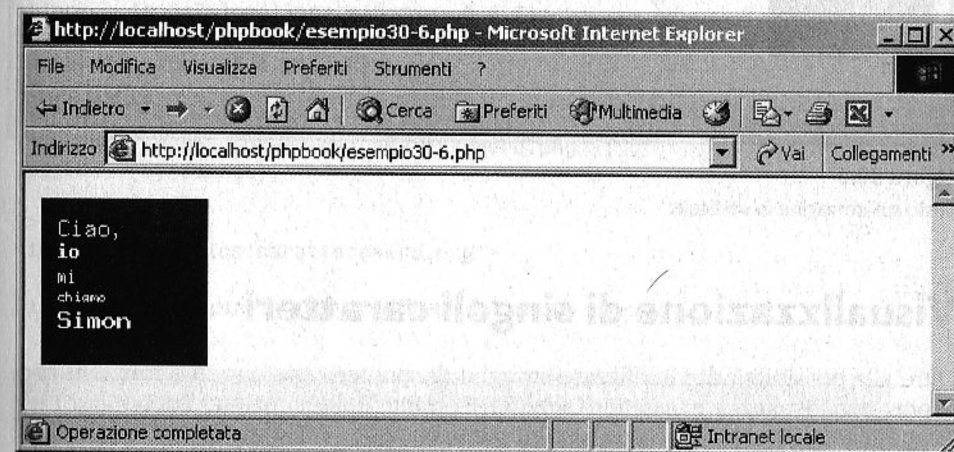


Figura 30.7
Testo sull'immagine.

Come si è detto in precedenza, la funzione `ImageStringUp()` è simile a `ImageString()`; infatti i parametri richiesti sono esattamente gli stessi. Lo script seguente illustra un esempio di utilizzo di questa funzione:

```
<?php

// Immagini GD - Esempio 30-7
//---------------------

$image = imagecreate(100,100);
$darkred = ImageColorAllocate($image,192,0,0);
```

```
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$darkred);
ImageStringUp($image,4,20,90,"Ciao", $white);
ImageStringUp($image,4,40,70,"Simon", $white);
ImagePNG($image, 'graphics/testorossovert.png');
ImageDestroy($image);

?>
<img src='graphics/testorossovert.png'>
```

La Figura 30.8 illustra l'output prodotto da questo script.

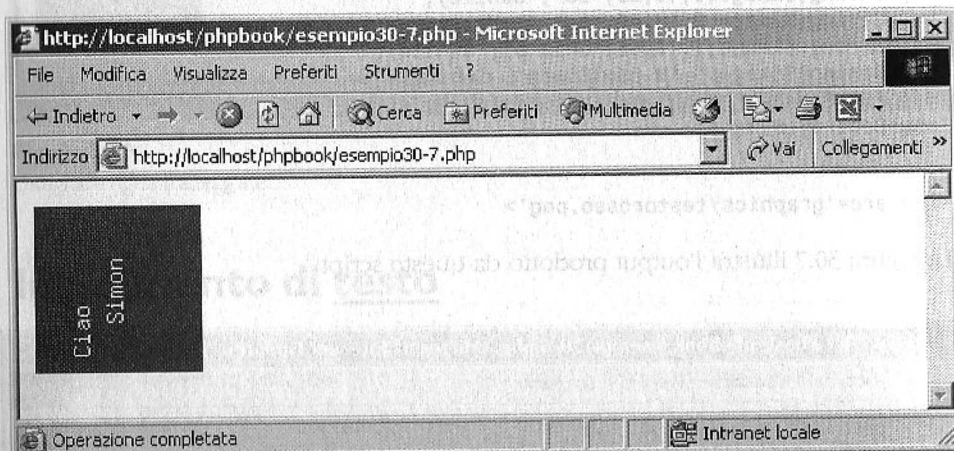


Figura 30.8
Testo sull'immagine in verticale.

Visualizzazione di singoli caratteri

Oltre alla possibilità di visualizzare stringhe di caratteri, esistono due funzioni concepite per visualizzare caratteri singoli. Si tratta delle funzioni `ImageChar()` e `ImageCharUp()`, che operano esattamente come le funzioni `ImageString()` e `ImageStringUp()`.
La sintassi delle funzioni è riportata di seguito.

```
Int ImageChar(int immagine, int dimensione, int x, int y, char carattere,
int colore);
Int ImageCharUp(int immagine, int dimensione, int x, int y, char
carattere, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
dimensione	int	La dimensione del carattere da visualizzare.
x	int	La coordinata X dell'angolo superiore sinistro del carattere.
y	int	La coordinata Y dell'angolo superiore sinistro del carattere.
carattere	char	Il carattere da visualizzare.
colore	int	Colore del rettangolo.
Restituzione di `imageChar()` e `imageCharUp()`	int	Restituisce `1` (`TRUE`) o `0` (`FALSE`) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageChar($image,4,20,70,"H", $white);
ImageCharUp($image,4,40,40,"S", $white);
```

Lo script seguente illustra un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-8
//----------------------

$image = imagecreate(100,100);
$black = ImageColorAllocate($image,0,0,0);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$black);
ImageChar($image,4,20,70,"H", $white);
ImageCharUp($image,4,40,40,"S", $white);
ImagePNG($image, 'graphics/caratterenero.png');
ImageDestroy($image);

?>
<img src='graphics/caratterenero.png'>
```

La Figura 30.9 illustra l'output prodotto da questo script.

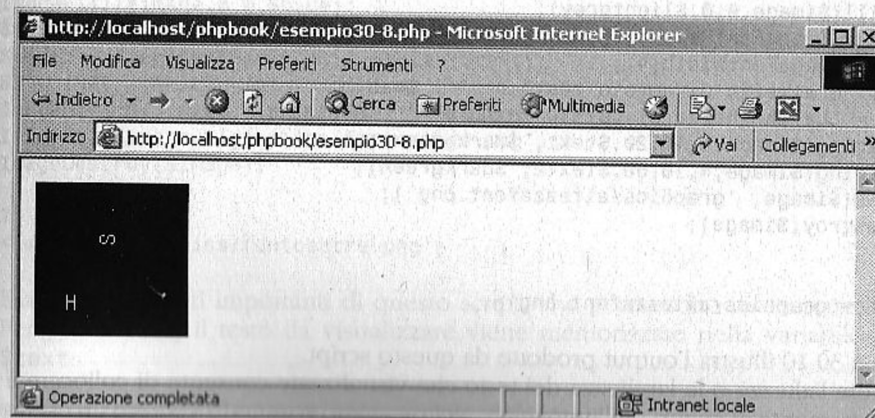


Figura 30.9
Visualizzazione di singoli caratteri.

Altezza e larghezza dei font

I font di diverse dimensioni, ovviamente, hanno larghezze e altezze diverse, e i font utilizzati nella libreria GD non fanno eccezione. La libreria GD ha due funzioni, utilizzabili per determinare la larghezza e l'altezza del font: `ImageFontWidth()` e `ImageFontHeight()`.
La sintassi delle funzioni è riportata di seguito.

```
int ImageFontWidth(int font);
int ImageFontHeight(int font);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
font	int	La dimensione del font che viene controllata, per determinarne l'equivalente in pixel.
Restituzione di imageFontWidth() e imageFontHeight()	int	Entrambe le funzioni restituiscono un valore intero, che rappresenta la larghezza o l'altezza del font in pixel.

Esempi di funzioni:

```
$height = ImageFontHeight(4);
$width = ImageFontWidth( 4);
```

Lo script seguente illustra un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-9
//----------------------

$image = imagecreate(300,100);
$lightgrey = ImageColorAllocate($image,192,192,255);
$darkgreen = ImageColorAllocate($image,0,192,0);
ImageFill($image,0,0,$lightgrey);
$height = ImageFontHeight(4);
$width = ImageFontWidth(4);
$text = "L'altezza di questo testo e' " . $width;
$text2 = "La larghezza di questo testo e' " . $height;
ImageString($image,4,10,20,$text, $darkgreen);
ImageString($image,4,10,50,$text2, $darkgreen);
ImagePNG($image, 'graphics/altezzafont.png');
ImageDestroy($image);

?>
<img src='graphics/altezzafont.png'>
```

La Figura 30.10 illustra l'output prodotto da questo script.
Conoscere l'altezza o la larghezza del testo che visualizzate consente di collocare il testo esattamente al centro dell'immagine. Nel prossimo paragrafo vedrete un esempio di quanto appena affermato.

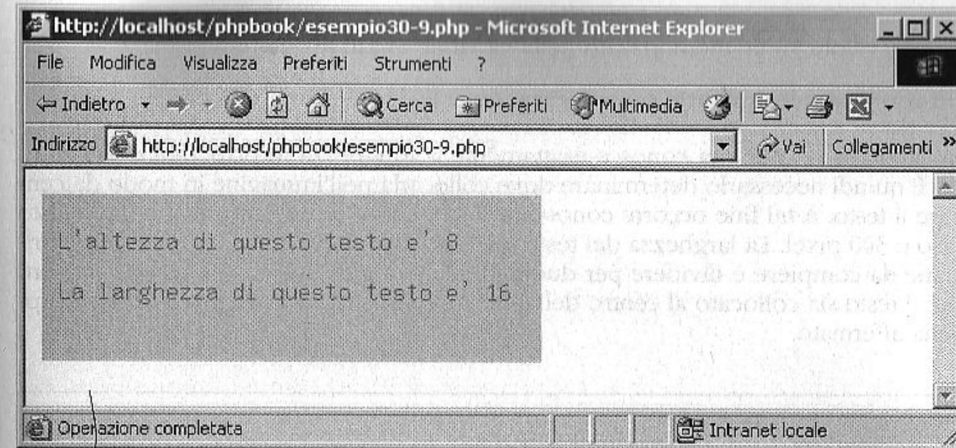


Figura 30.10
Visualizzazione della larghezza e dell'altezza del testo.

Testo centrato

Una frequente applicazione della funzione `ImageFontWidth()` è il posizionamento del testo esattamente al centro dell'immagine.
Considerate lo script seguente, che illustra come agire per centrare il testo.

```
<?php

// Immagini GD - Esempio 30-10
//----------------------

$image = imagecreate(300,100);
$blue = ImageColorAllocate($image,0,0,255);
$yellow = ImageColorAllocate($image,255,255,0);
ImageFill($image,0,0,$blue);
$text = "Esempio di testo centrato";
$width = ImageFontWidth(4) * strlen($text);
$x = (300 - $width)/2;
ImageString($image,4,$x,40,$text, $yellow);
ImagePNG($image, 'graphics/fontcentre.png');
ImageDestroy($image);

?>
<img src='graphics/fontcentre.png'>
```

Esaminate le parti importanti di questo script, che permettono di centrare il testo. Per prima cosa, il testo da visualizzare viene memorizzato nella variabile stringa `$text`:

```
$text = "Esempio di testo centrato";
```

Ovviamente potete utilizzare il testo che preferite. La riga successiva calcola la larghezza del font con la funzione `ImageFontWidth()` e moltiplica questo valore per

il numero di caratteri presenti nella stringa, ottenuto con la funzione `strlen()`. Questo valore viene memorizzato nella variabile `$width`:

```
$width = ImageFontWidth(4) * strlen($text);
```

La variabile `$width` ora conosce esattamente la lunghezza in pixel dell'intera stringa. È quindi necessario determinare dove collocarla nell'immagine in modo da centrare il testo. A tal fine occorre conoscere la larghezza dell'immagine, che in questo caso è 300 pixel. La larghezza del testo già è nota, quindi l'unica operazione che rimane da compiere è dividere per due la differenza tra i due valori, per essere certi che il testo sia collocato al centro dell'immagine. La Figura 30.11 illustra quanto appena affermato.

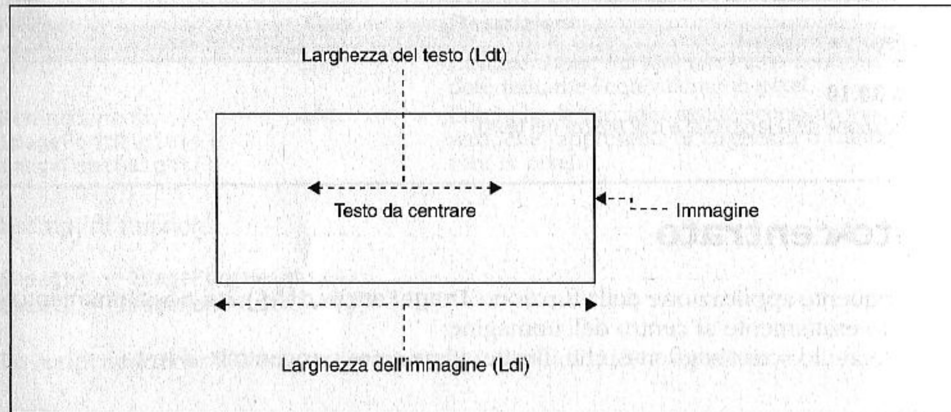


Figura 30.11
Testo centrato.

Per calcolare la posizione centrata si sottrae la larghezza del testo (Ldt) dalla larghezza dell'immagine (Ldi) e si divide il risultato per due. Il codice per calcolare la posizione della coordinata X è il seguente:

```
$x = (300 - $width)/2;
```

La Figura 30.12 illustra l'output prodotto da questo programma.

Creazione di poligoni

Oltre a semplici rettangoli e quadrati, la libreria GD offre il supporto per la creazione di poligoni più complessi. A tal fine si utilizza la funzione `ImagePolygon()`. La sintassi della funzione è la seguente:

```
int ImagePolygon(int immagine, array punti, int numero, int colore);
```




Figura 30.12
Testo centrato.

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	`int`	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
punti	`array`	L'array che memorizza i punti dell'immagine.
numero	`int`	Il numero di punti che realizzano il poligono.
colore	`int`	Il colore del poligono.
Restituzione di `imagePolygon()`	`int`	Restituisce `1` (`TRUE`) o `0` (`FALSE`) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
$coords =array(0,50,35,35,50,0,65,35,100,50,65,65,50,100,35,65);
ImagePolygon($image,$coords,8,$yellow);
```

In questo esempio il poligono è costituito da otto punti che sono memorizzati nell'array `$coords` come otto coppie di valori X-Y. Lo script che segue illustra un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-11
//---------------------

$image = imagecreate(100,100);
$blue = ImageColorAllocate($image,0,0,255);
$yellow = ImageColorAllocate($image,255,255,0);
ImageFill($image,0,0,$blue);

$coords = array(0,50,35,35,50,0,65,35,100,50,65,65,50,100,35,65);
ImagePolygon($image,$coords,8,$yellow);
ImagePNG($image, 'graphics/stella.png');
ImageDestroy($image);
```

```
?>
<img src='graphics/stella.png'>
```

La Figura 30.13 illustra l'output prodotto da questo script.

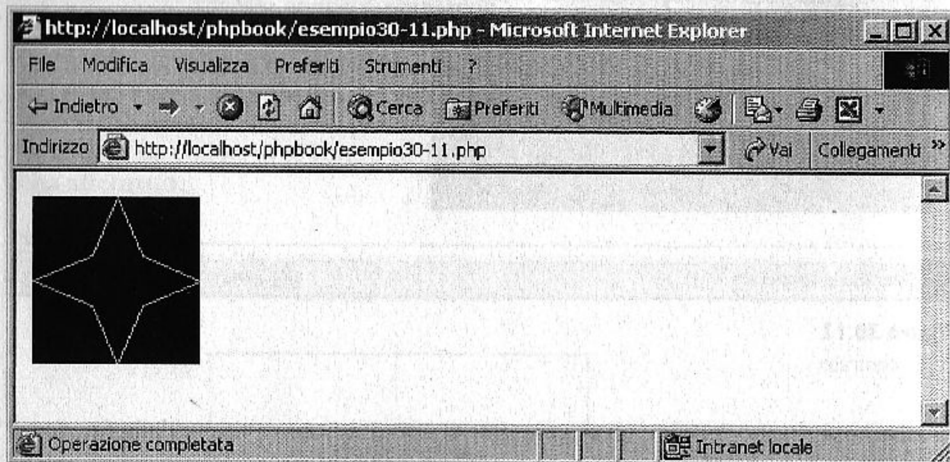


Figura 30.13
Creazione di un poligono.

Riempimento di forme

Potreste voler creare un poligono come quello dell'esempio precedente che non sia un semplice contorno, ma una figura piena. Sfortunatamente non esiste una funzione `ImagePolygonFilled()`, tuttavia avete a disposizione qualcosa di molto simile: la funzione `ImageFillToBorder()`. Questa funzione consente di riempire parte di un'immagine con un colore. Un cambiamento del colore di sfondo dell'immagine indica in che misura avverrà il riempimento di colore.
La sintassi della funzione è la seguente.

```
Int ImageFilledToBorder(int immagine, int x, int y, int coloreBordo, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
x	int	La coordinata X dell'inizio del riempimento.
y	int	La coordinata Y dell'inizio del riempimento.
coloreBordo	int	Il colore del bordo in corrispondenza del quale interrompere il riempimento.
colore	int	Colore del riempimento.
Restituzione di `imageFilledToBorder()`	int	Restituisce 1 (`TRUE`) o 0 (`FALSE`) a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageFillToBorder($image,50,50,$yellow,$red);
```

Osservate che in questo esempio il poligono verrà riempito con il colore giallo, mentre il bordo del poligono sarà rosso. Lo script che segue illustra un esempio di utilizzo di questa funzione.

```
<?php

// Immagini GD - Esempio 30-12
//----------------------

$image = imagecreate(100,100);
$blue = ImageColorAllocate($image,0,0,255);
$yellow = ImageColorAllocate($image,255,255,0);
$red = ImageColorAllocate($image,255,0,0);
ImageFill($image,0,0,$blue);

$coords = array(0,50,35,35,50,0,65,35,100,50,65,65,50,100,35,65);
ImagePolygon($image,$coords,8,$yellow);
ImageFillToBorder($image,50,50,$yellow,$red);
ImagePNG($image, 'graphics/stellariempita.png');
ImageDestroy($image);

?>
<img src='graphics/stellariempita.png'>
```

La Figura 30.14 illustra l'output prodotto da questo script.

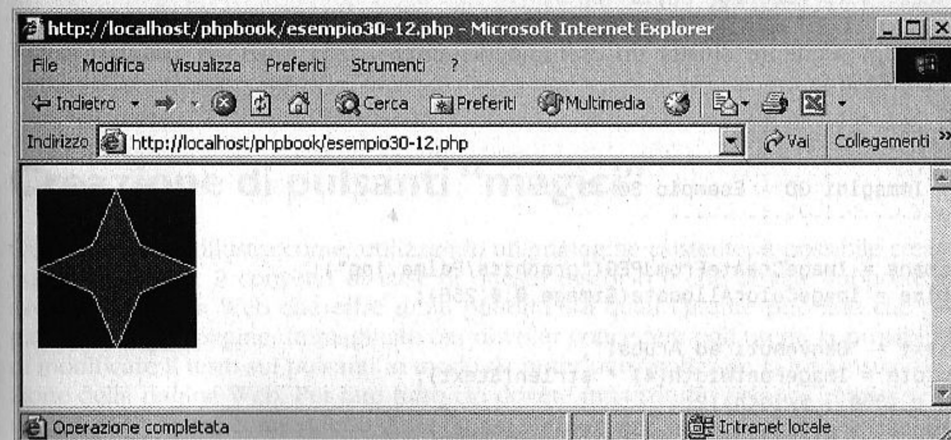


Figura 30.14
Poligono riempito.

Creazione di nuove immagini a partire da quelle esistenti

Quando utilizzate la libreria GD, non siete costretti a cominciare a creare le immagini da zero: potete anche caricare un'immagine esistente e modificarla. PHP offre il supporto per tre formati di immagine diversi (JPEG, GIF e PNG) rispettivamente con le funzioni `ImageCreateFromJPEG()`, `ImageCreateFromGIF()` e `ImageCreateFromPNG()`.
La sintassi delle funzioni è riportata di seguito.

```
int ImageCreateFromJPEG(string nomefile);
int ImageCreateFromGIF(string nomefile);
int ImageCreateFromPNG(string nomefile);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
nomefile	string	Il nome del file di immagine che funge da base per la nuova immagine.
Restituzione di imageCreateFromGIF() imageCreateFromPNG() imageCreateFromJPEG()	int	La funzione restituisce un valore intero, che viene utilizzato per memorizzare l'handle dell'immagine, proprio come avviene con la funzione ImageCreate().

Esempio di funzione:

```
$image = ImageCreateFromJPEG("palma.jpg");$image =
ImageCreateFromJPEG("palma.jpg");
```

Lo script seguente illustra un esempio di utilizzo della funzione `ImageCreateFromJPEG()`.

```
<?php

// Immagini GD - Esempio 30-13
//----------------------

$image = ImageCreateFromJPEG("graphics/Palma.jpg");
$blue = ImageColorAllocate($image,0,0,255);

$text = "Benvenuti ad Aruba!";
$width = ImageFontWidth(4) * strlen($text);
$x = (300 - $width)/2;
ImageString($image,4,$x,330,$text, $blue);
ImagePNG($image, 'graphics/palma.png');
ImageDestroy($image);

?>
<img src='graphics/palma.png'>
```

La Figura 30.15 illustra l'output prodotto da questo script.
Ma ora che avete visto che è possibile modificare un'immagine esistente, come potete sfruttare questa capacità? Troverete la risposta nel prossimo paragrafo.

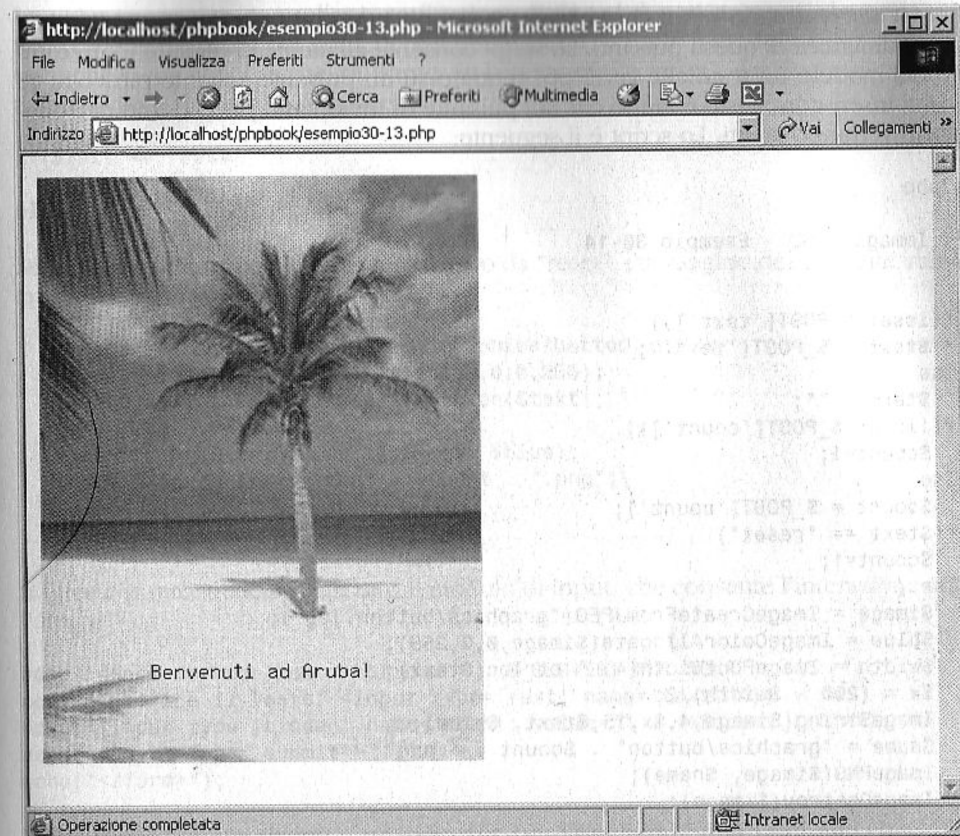


Figura 30.15
Modifica di un'immagine esistente.

Creazione di pulsanti "magici"

Questo esempio illustra come, utilizzando un'immagine esistente, è possibile creare nuove immagini. Il concetto di base di questo discorso è che dovete supporre di avere una pagina Web che offre molti pulsanti sui quali l'utente può fare clic per passare ad altre pagine. Immaginate ora di voler concedere agli utenti la possibilità di modificare il testo sui pulsanti in modo da poter personalizzare la loro visualizzazione della pagina Web. Per fare tutto ciò dovete innanzitutto creare l'immagine di un pulsante vuoto, come quello della Figura 30.16.

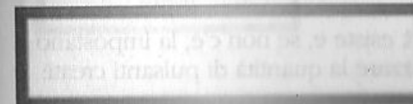


Figura 30.16
Pulsante vuoto.

Poi viene creato uno script che utilizza un semplice modulo per dimostrare la creazione dinamica di questi pulsanti. Lo script consente all'utente di digitare in un campo del modulo un testo che verrà poi centrato su un pulsante. Quante più volte gli utenti inseriscono testo e fanno clic sul pulsante di invio, tanti più pulsanti personalizzati verranno creati. Lo script è il seguente:

```
<?php

// Immagini GD - Esempio 30-14
//---------------------

if(isset($_POST['text']))
   $text = $_POST['text'];
else
   $text = "";
if(!isset($_POST['count']))
   $count=1;
else
   $count = $_POST['count'];
if($text == "reset")
   $count=1;
else {
   $image = ImageCreateFromJPEG("graphics/button.jpg");
   $blue = ImageColorAllocate($image,0,0,255);
   $width = ImageFontWidth(4) * strlen($text);
   $x = (200 - $width)/2;
   ImageString($image,4,$x,15,$text, $blue);
   $name = "graphics/button" . $count . ".png";
   ImagePNG($image, $name);
   ImageDestroy($image);
}
echo("<form method='post' action='" . $_SERVER['PHP_SELF'] . "'>");
echo("Inserire il testo: <input type='text' name='text'>");
echo("<input type='hidden' name='count' value='" . ++$count . "'>");
echo("<input type='submit'>");
echo("</form>");
for($a=2;$a<$count;$a++)
   echo("<img src='graphics/button" . $a . ".png'>");

?>
```

Esaminate lo script in modo più dettagliato. Le prime righe controllano il valore del testo utilizzato per memorizzare il testo del pulsante:

```
if(isset($_POST['text']))
   $text = $_POST['text'];
else
   $text = "";
```

Le righe successive verificano se la variabile `$count` esiste e, se non c'è, la impostano a 1. La variabile `$count` viene impiegata per memorizzare la quantità di pulsanti creati:

```
if(!isset($_POST['count']))
   $count=1;
else
   $count = $_POST['count'];
```


Le righe successive controllano se il valore di `$text` è pari a "reset". La variabile `$text` viene utilizzata per contenere il testo inserito tramite il modulo e per creare il testo che viene collocato nel pulsante. Se l'utente digita "reset", il numero di pulsanti creati viene riportato a 1, consentendo di ricominciare:

```
if($text == "reset")
    $count=1;
else {
```

Se il valore della variabile `$text` è diverso da "reset", l'immagine del pulsante viene creata:

```
   $image = ImageCreateFromJPEG("graphics/button.jpg");
   $blue = ImageColorAllocate($image,0,0,255);
   $width = ImageFontWidth(4) * strlen($text);
   $x = (200 - $width)/2;
   ImageString($image,4,$x,15,$text, $blue);
   $name = "graphics/button" . $count . ".png";
   ImagePNG($image, $name);
   ImageDestroy($image);
```

A questo punto viene visualizzato il modulo di input, che consente l'interazione con l'utente:

```
echo("<form method='post' action='" . $_SERVER['PHP_SELF'] . "'>");
echo("Inserire il testo: <input type='text' name='text'>");
echo("<input type='hidden' name='count' value='" . ++$count . "'>");
echo("<input type='submit'>");
echo("</form>");
```

Infine il programma ricorre a un ciclo `for` per visualizzare tutti i pulsanti che sono stati creati:

```
for($a=2;$a<$count;$a++)
   echo("<img src='graphics/button" . $a . ".png'>");
```

La Figura 30.17 illustra un esempio di output prodotto da questo script.

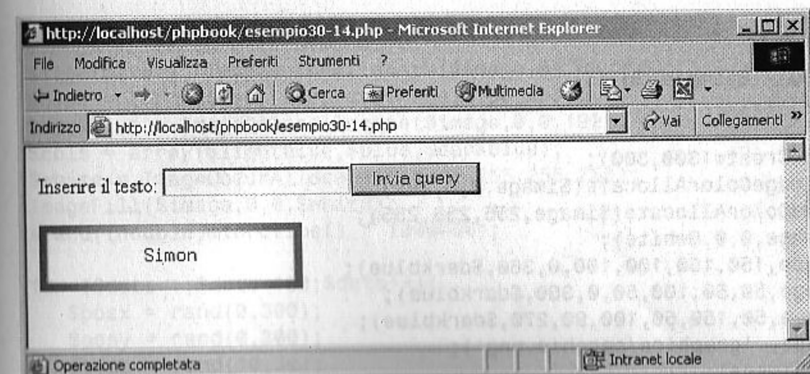


Figura 30.17
Output dei pulsanti magici.

Utilizzo di cerchi, ellissi e archi

La funzione `ImageArc()` può essere impiegata per produrre cerchi, ellissi, semicerchi e archi.
La sintassi della funzione è la seguente.

```
int ImageArc(int immagine, int centroX, int centroY, int larghezza, int
altezza, int gradiInizio, int gradiFine, int colore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
immagine	int	Handle dell'immagine ottenuto dalla funzione `ImageCreate()`.
centroX	int	La coordinata X del centro del cerchio o dell'ellisse.
centroY	int	La coordinata Y del centro del cerchio o dell'ellisse.
larghezza	int	Larghezza del cerchio o dell'ellisse.
altezza	int	Altezza del cerchio o dell'ellisse.
gradiInizio	int	Inizia a disegnare da questi gradi.
gradiFine	int	Finisce di disegnare a questi gradi.
colore	int	Il colore del cerchio o dell'ellisse.
Restituzione di `imageArc()`	int	Restituisce `1 (TRUE)` o `0 (FALSE)` a seconda che la funzione abbia avuto o meno successo.

Esempio di funzione:

```
ImageArc($image,150,150,100,100,0,360,$darkblue);
```

Quando si disegnano cerchi, ellissi o archi, è necessario specificare l'angolo di inizio (in gradi) e quello di termine. Immaginate di voler disegnare un cerchio completo. Ebbene, l'angolo iniziale sarebbe 0 gradi e quello finale sarebbe 360 gradi, in quanto ciò che volete è un cerchio completo. Qualora invece vogliate disegnare un semicerchio, potreste specificare 90 gradi come angolo iniziale e 270 come angolo finale. Lo script che segue illustra un esempio di utilizzo della funzione `ImageArc()`.

```
<?php

// Immagini GD - Esempio 30-15
//-----------------------

$image = ImageCreate(300,300);
$darkblue = ImageColorAllocate($image,0,0,192);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$white);
ImageArc($image,150,150,100,100,0,360,$darkblue);
ImageArc($image,50,50,100,50,0,360,$darkblue);
ImageArc($image,50,150,50,100,90,270,$darkblue);
ImagePNG($image, 'graphics/cerchio.png');
ImageDestroy($image);

?>
<img src='graphics/cerchio.png'>
```

La Figura 30.18 mostra un esempio di output prodotto da questo script.

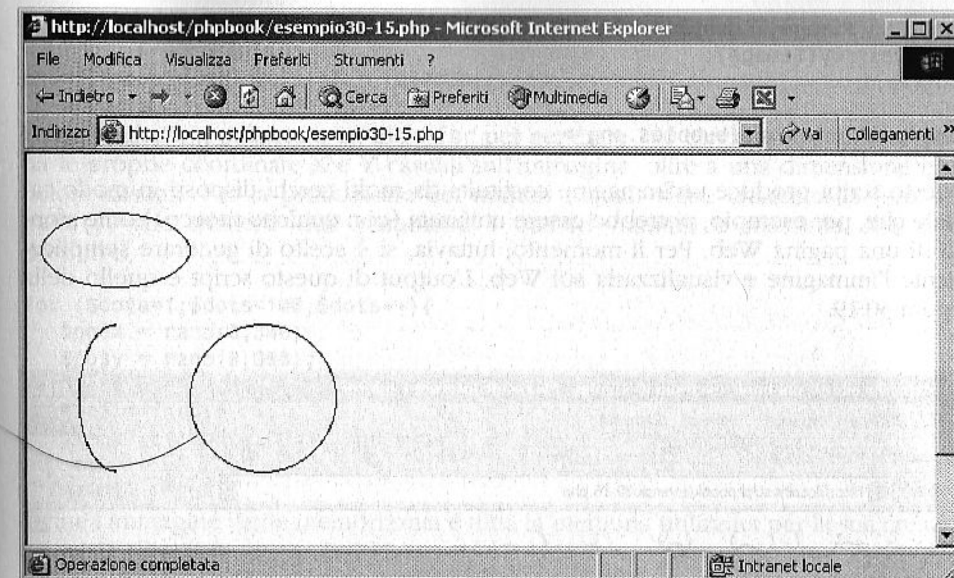


Figura 30.18
Visualizzazione di cerchi, ellissi e archi.

Produzione di "bolle" casuali

La funzione `ImageArc()` può essere utilizzata per produrre effetti molto interessanti. Lo script che segue illustra un semplice esempio.

```
<?php

// Immagini GD - Esempio 30-16
//-----------------------

$image = ImageCreate(300,300);
$lightblue =  ImageColorAllocate($image,64,64,255);
$blue = ImageColorAllocate($image,0,50,255);
$darkblue = ImageColorAllocate($image,0,0,192);
$cols = array($lightblue,$blue,$darkblue);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$white);
srand((double)microtime() * 1000000);

for ($dots=1;$dots<100;$dots++){
   $posx = rand(0,300);
   $posy = rand(0,300);
   $size = rand(10,30);
   $col = rand(0,2);
```

```
  ImageArc($image,$posx,$posy,$size,$size,0,360,$cols[$col]);
}

ImagePNG($image, 'graphics/bubbles.png');
ImageDestroy($image);

?>
<img src='graphics/bubbles.png'>
```

Questo script produce un'immagine costituita da molti cerchi disposti in modo casuale che, per esempio, potrebbe essere utilizzata (con qualche ritocco) come sfondo di una pagina Web. Per il momento, tuttavia, si è scelto di generare semplicemente l'immagine e visualizzarla sul Web. L'output di questo script è quello della Figura 30.19.

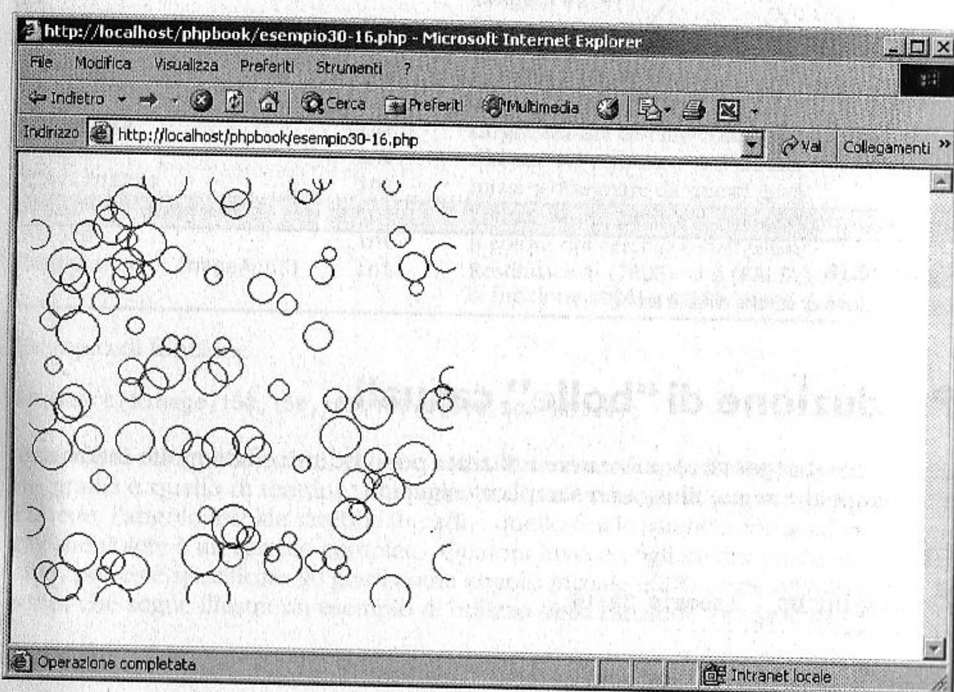


Figura 30.19
Cerchi casuali.

Osservate il funzionamento del programma. Le prime righe del codice dovrebbero ormai esservi familiari, in quanto si limitano a creare un'immagine di 300 × 300 pixel, definiscono alcuni colori che verranno applicati e riempiono l'immagine con il bianco.

```
$image = ImageCreate(300,300);
$lightblue =  ImageColorAllocate($image,64,64,255);
$blue = ImageColorAllocate($image,0,50,255);
$darkblue = ImageColorAllocate($image,0,0,192);
$cols = array($lightblue,$blue,$darkblue);
```



```
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$white);
```

La riga successiva viene utilizzata per inizializzare il generatore di numeri casuali:

```
srand((double)microtime() * 1000000);
```

Le righe ulteriori utilizzano un ciclo `for` per generare 100 cerchi, ciascuno dei quali ha le proprie coordinate X e Y casuali sull'immagine, oltre a una dimensione e un colore casuali. Per la produzione dei numeri casuali viene chiamata la funzione `rand()`, che richiede due parametri, i numeri minimo e massimo che deve produrre:

```
for ($dots=1;$dots<100;$dots++){
  $posx = rand(0,300);
  $posy = rand(0,300);
  $size = rand(10,30);
  $col = rand(0,2);
  ImageArc($image,$posx,$posy,$size,$size,0,360,$cols[$col]);
}
```

Infine l'immagine viene memorizzata e tutta la memoria utilizzata per la sua creazione viene rilasciata:

```
ImagePNG($image, 'graphics/bubbles.png');
ImageDestroy($image);
```

Motivi ripetuti

Questo paragrafo mostra che cosa è possibile produrre con un semplice programma, un numero casuale e un insieme di regole di base. Lo script seguente non introduce funzioni o concetti nuovi: è una semplice dimostrazione di ciò che è possibile produrre.

```
<?php

// Immagini GD - Esempio 30-17
//---------------------

$image = ImageCreate(251,251);
$blue = ImageColorAllocate($image,0,50,255);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$white);
srand((double)microtime() * 1000000);

$x = array(125,250,0);
$y = array(0,250,250);

ImageLine($image,$x[0],$y[0],$x[1],$y[1],$blue);
ImageLine($image,$x[1],$y[1],$x[2],$y[2],$blue);
ImageLine($image,$x[2],$y[2],$x[0],$y[0],$blue);

$lx=0;
$ly=0;
$count=0;
```

```
while ($count < 10000) {
   $count++;
   $rand = rand(0,2);
   $nx = ($x[$rand] - $lx) / 2;
   $ny = ($y[$rand] - $ly) / 2;
   ImageLine($image,$lx+$nx,$ly+$ny,$lx+$nx,$ly+$ny,$blue);
   $lx = $lx+$nx;
   $ly = $ly+$ny;
}

ImagePNG($image, 'graphics/triangolo.png');
ImageDestroy($image);

?>
<img src='graphics/triangolo.png'>
```

L'output prodotto da questo programma è quello della Figura 30.20.

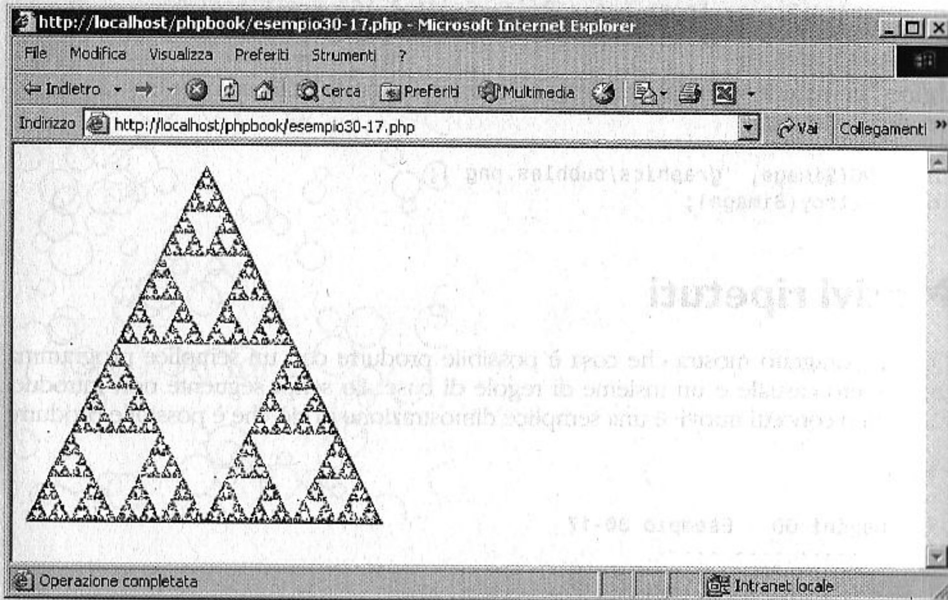


Figura 30.20
Motivi ripetuti.

Se volete scoprite come si forma l'immagine della Figura 30.20, dovete esaminare questo script in modo più dettagliato. Le prime righe di codice definiscono un'immagine di 251 × 251 pixel e creano alcuni colori.
L'immagine viene riempita di bianco e il generatore di numeri casuali viene inizializzato:

```
$image = ImageCreate(251,251);
$blue = ImageColorAllocate($image,0,50,255);
$white = ImageColorAllocate($image,255,255,255);
ImageFill($image,0,0,$white);
srand((double)microtime() * 1000000);
```

Poi vengono creati due array di coordinate X e Y e il loro contenuto viene utilizzato da tre funzioni `ImageLine()` per la creazione del triangolo di contorno dell'immagine:

```
$x = array(125,250,0);
$y = array(0,250,250);

ImageLine($image,$x[0],$y[0],$x[1],$y[1],$blue);
ImageLine($image,$x[1],$y[1],$x[2],$y[2],$blue);
ImageLine($image,$x[2],$y[2],$x[0],$y[0],$blue);
```

Quindi viene impiegato un ciclo `while` per generare una riga (della lunghezza effettiva di un solo pixel) sull'immagine. Questo induce la creazione del motivo geometrico:

```
$lx=0;
$ly=0;
$count=0;
while ($count < 10000) {
   $count++;
   $rand = rand(0,2);
   $nx = ($x[$rand] - $lx) / 2;
   $ny = ($y[$rand] - $ly) / 2;
   ImageLine($image,$lx+$nx,$ly+$ny,$lx+$nx,$ly+$ny,$blue);
   $lx = $lx+$nx;
   $ly = $ly+$ny;
}
```

Infine, l'immagine viene archiviata e la memoria rilasciata:

```
ImagePNG($image, 'graphics/triangolo.png');
ImageDestroy($image);
```

Riepilogo

Questo capitolo ha introdotto la libreria GD e ha mostrato come le immagini possano essere create da zero e come sia possibile modificare immagini già esistenti. Una volta comprese le basi della libreria di funzioni, dovreste riuscire a produrre anche immagini molto complesse. Nel capitolo sono stati offerti alcuni esempi degli effetti che è possibile produrre con la libreria, mentre nel prossimo si vedrà come sfruttare le tecniche e le funzioni illustrare in queste pagine per creare immagini grafiche dinamiche particolarmente utili.

Capitolo 31

Creazione di un'immagine dinamica

Introduzione

Nel capitolo precedente è stata introdotta la libreria GD e sono state presentate alcune tra le numerose funzioni disponibili per la creazione e la manipolazione di immagini. È stato anche proposto qualche esempio di utilizzo della libreria, mettendo in luce in particolare come sfruttare le funzioni per lo sviluppo di pagine Web dinamiche.
Questo capitolo presenta un'applicazione completa che utilizza molte funzioni di libreria analizzate nel capitolo precedente.
Si inizierà presentando l'applicazione e fornendo qualche esempio delle sue potenzialità; sarete quindi guidati nell'analisi dello script e scoprirete come è stata sviluppata l'applicazione.

Presentazione dell'immagine

Lo script può creare semplici istogrammi: nella Figura 31.1 potete vedere un esempio di quello che si può ottenere con esso. L'immagine è costruita con la chiamata a una funzione `graph()`; si tratta di una funzione configurabile e questo consente all'utente di stabilire la dimensione reale dell'istogramma, i colori utilizzati, il testo dei titoli e delle etichette, nonché i valori dei dati visualizzati. Tutto ciò significa che la grande versatilità di questa funzione consente di utilizzarla per visualizzare in modo dinamico una vasta gamma di tipi di dati nei siti Web.
Magari volete rappresentare graficamente il numero dei nuovi utenti che accedono mese per mese al sito, oppure le risposte degli utenti del sito Web a una serie di domande tratte da un questionario.
Anche se i dati visualizzati in questo grafico sono statici, nei prossimi capitoli imparerete a interfacciare PHP con i database. In tal modo potrete ottenere i dati mediante un modulo Web, memorizzarli in un database e poi recuperare e visualizzare questi dati con la funzione `graph()`.
Per ora limitatevi ad analizzare la schermata di esempio della Figura 31.1. Essa mostra un grafico che visualizza 12 valori mensili. La dimensione del grafico è di 300 × 200 pixel. La scala del grafico è calcolata automaticamente e ha un intervallo 0-100.

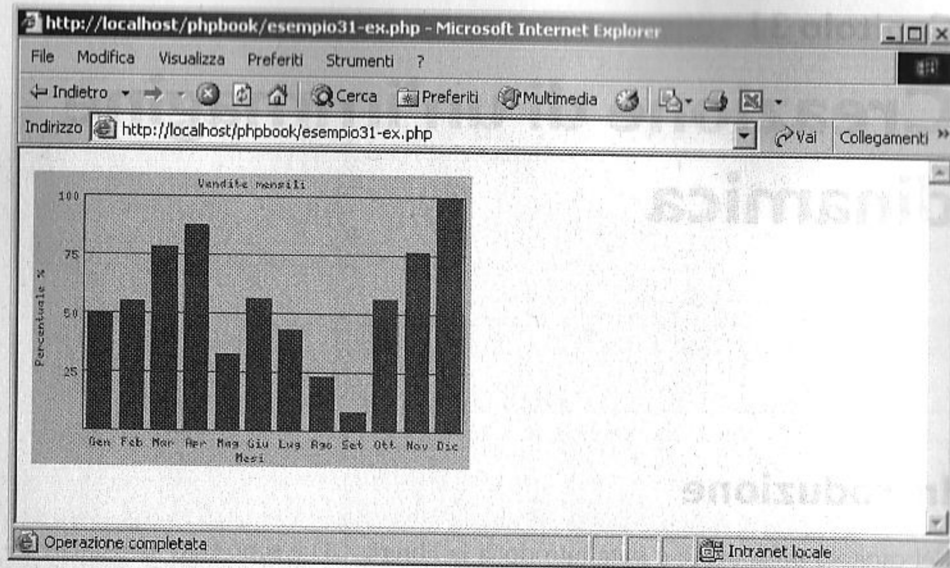


Figura 31.1
Istogramma con valori mensili, esempio 1.

La Figura 31.2 mostra un altro grafico realizzato con la stessa funzione `graph()`. In questo secondo esempio sono riportati cinque valori che rappresentano il livello di qualità di un servizio. La dimensione del grafico è 250 × 200 pixel e, ancora una volta, la scala del grafico viene creata automaticamente: 0-1456.

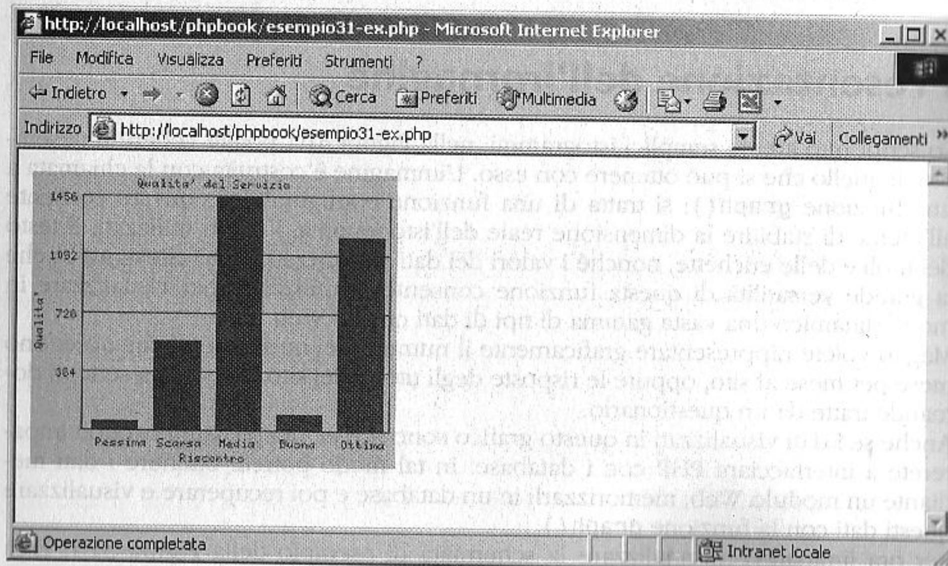


Figura 31.2
Istogramma, esempio 2.

Infine, la Figura 31.3 presenta un ulteriore esempio di grafico a barre, costituito da due valori dati. Questi valori rappresentano le risposte a domande del tipo "sì/no".

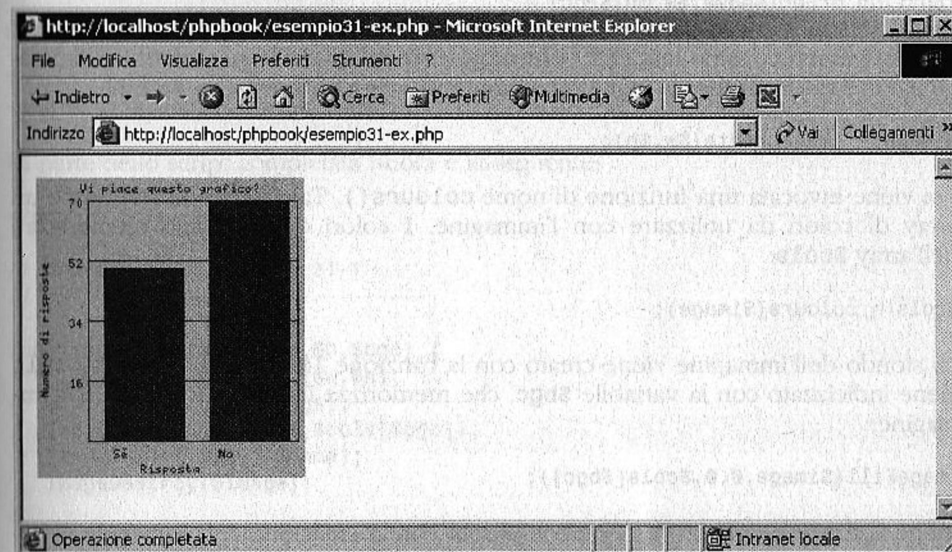


Figura 31.3
Istogramma, esempio 3.

Tutti e tre gli esempi dimostrano che lo sviluppatore controlla la dimensione dell'immagine del grafico, i colori impiegati, le etichette visualizzate e i valori dei dati utilizzati per creare le barre.
Il capitolo prosegue analizzando la creazione di un grafico con la funzione `graph()`. Il progetto finale è stato suddiviso in parti più piccole e più comprensibili che verranno esaminate una alla volta, fino ad ottenere la creazione dell'applicazione completa.

Creazione di uno sfondo colorato ridimensionabile

Lo sviluppo del grafico ha inizio progettando le parti della funzione che permettono di creare l'immagine e colorarla. La funzione `graph()` inizialmente appare come segue:

```
<?php

function graph($name,$w,$h,$bgc) {
   $image = ImageCreate($w,$h);
   $cols = colours($image);
   ImageFill($image,0,0,$cols[$bgc]);
   ImageJPEG($image, $name);
   ImageDestroy($image);
}
```

La funzione è stata scritta in modo da ricevere quattro parametri: il nome dell'immagine da creare, la larghezza e l'altezza dell'immagine, il colore dello sfondo:

```
function graph($name,$w,$h,$bgc) {
```

Successivamente, viene invocata la funzione `ImageCreate()` per costruire l'immagine con i valori larghezza e altezza memorizzati rispettivamente in `$w` e `$h`:

```
$image = ImageCreate($w,$h);
```

Poi viene invocata una funzione di nome `colours()`. Tale funzione restituisce un array di colori da utilizzare con l'immagine. I colori definiti sono memorizzati nell'array `$cols`:

```
$cols = colours($image);
```

Lo sfondo dell'immagine viene creato con la funzione `ImageFill()`. L'array `cols` viene indicizzato con la variabile `$bgc`, che memorizza il colore di sfondo dell'immagine:

```
ImageFill($image,0,0,$cols[$bgc]);
```

Infine la funzione `graph()` viene completata con la creazione dell'immagine utilizzando il valore memorizzato in `$name`, quindi la memoria viene rilasciata:

```
ImageJPEG($image, $name);
ImageDestroy($image);
```

La funzione `colours()` viene utilizzata per creare un array contenente le definizioni dei colori da applicare all'immagine. Specificando un valore compreso tra 0 e 14, l'utente può fare riferimento a un particolare colore memorizzato nell'array:

```
function colours($image) {
   $white = ImageColorAllocate($image,255,255,255);
   $black = ImageColorAllocate($image,0,0,0);
   $lightblue =  ImageColorAllocate($image,64,64,255);
   $blue = ImageColorAllocate($image,0,0,255);
   $darkblue = ImageColorAllocate($image,0,0,192);
   $red = ImageColorAllocate($image,255,0,0);
   $lightred = ImageColorAllocate($image,255,64,64);
   $darkred = ImageColorAllocate($image,192,0,0);
   $green = ImageColorAllocate($image,0,255,0);
   $lightgreen = ImageColorAllocate($image,64,255,64);
   $darkgreen = ImageColorAllocate($image,0,192,0);
   $yellow = ImageColorAllocate($image,255,255,0);
   $grey = ImageColorAllocate($image,192,192,192);
   $darkgrey = ImageColorAllocate($image,128,128,128);
   $lightgrey = ImageColorAllocate($image,192,192,255);
   $cols = array($white,$black,$lightblue,$blue,$darkblue,$red,
                 $lightred,$darkred,$green,$lightgreen,$darkgreen,
                 $yellow,$grey,$darkgrey,$lightgrey);
   return $cols;
}
```

La funzione `graph()` viene invocata con una semplice chiamata. I valori passati alla funzione sono: `"graph.jpeg"` (il nome dell'immagine) **200** (la larghezza) **150** (l'altezza) e **14** (il colore di sfondo che corrisponde a grigio chiaro):

```
graph("graphics/graph.jpeg", 200, 150, 14);

?>
<img src='graphics/graph.jpeg'>
```

La parte dello script completata finora è la seguente:

```
<?php

// Grafico GD - Esempio 31-1
//-------------------------------

function graph($name,$w,$h,$bgc) {
   $image = ImageCreate($w,$h);
   $cols = colours($image);
   ImageFill($image,0,0,$cols[$bgc]);
   ImageJPEG($image, $name);
   ImageDestroy($image);
}

function colours($image) {
   $white = ImageColorAllocate($image,255,255,255);
   $black = ImageColorAllocate($image,0,0,0);
   $lightblue =  ImageColorAllocate($image,64,64,255);
   $blue = ImageColorAllocate($image,0,0,255);
   $darkblue = ImageColorAllocate($image,0,0,192);
   $red = ImageColorAllocate($image,255,0,0);
   $lightred = ImageColorAllocate($image,255,64,64);
   $darkred = ImageColorAllocate($image,192,0,0);
   $green = ImageColorAllocate($image,0,255,0);
   $lightgreen = ImageColorAllocate($image,64,255,64);
   $darkgreen = ImageColorAllocate($image,0,192,0);
   $yellow = ImageColorAllocate($image,255,255,0);
   $grey = ImageColorAllocate($image,192,192,192);
   $darkgrey = ImageColorAllocate($image,128,128,128);
   $lightgrey = ImageColorAllocate($image,192,192,255);
   $cols = array($white,$black,$lightblue,$blue,$darkblue,$red,
                 $lightred,$darkred,$green,$lightgreen,$darkgreen,
                 $yellow,$grey,$darkgrey,$lightgrey);
   return $cols;
}

graph("graphics/graph.jpeg", 200, 150, 14);

?>
<img src='graphics/graph.jpeg'>
```

L'output ottenuto da questo programma è quello della Figura 31.4.

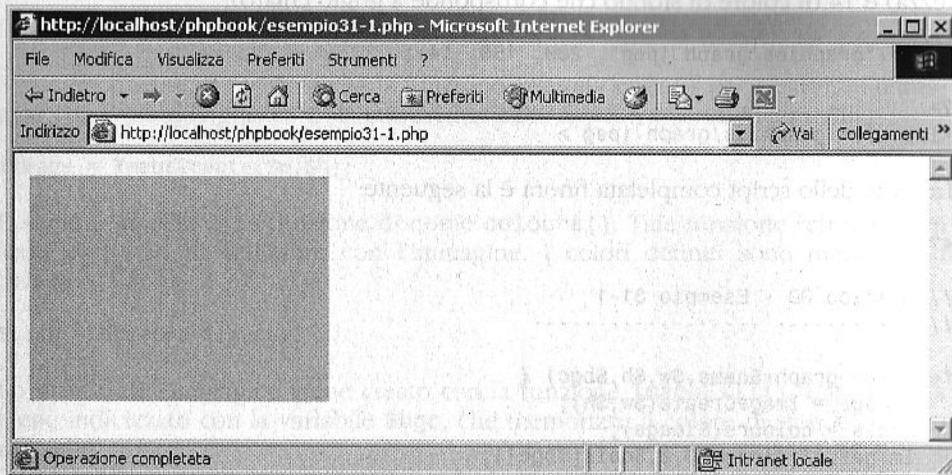


Figura 31.4
Grafico parte 1, immagine dimensionabile.

Creazione dell'asse del grafico e delle righe orizzontali

La parte successiva dello script si occupa della creazione di un bordo che contrassegni gli assi dell'istogramma e di qualche riga orizzontale. Le prime modifiche apportate allo script riguardano la funzione `graph()`, il cui numero di parametri è stato aumentato per consentire all'utente di specificare il colore di primo piano:

```
function graph($name,$w,$h,$bgc,$fgc) {
```

All'inizio della funzione `graph()` vengono aggiunte quattro variabili per definire i bordi superiore, inferiore, destro e sinistro del grafico:

```
$top=15;
$right=5;
$left=35;
$bottom=25;
```

Quindi, dopo la chiamata alla funzione `ImageFill()` all'interno della funzione `graph()`, vengono inserite due nuove chiamate a funzione:

```
drawAxis($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
drawLines($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
```

Queste due funzioni creano l'asse del grafico e le linee tratteggiate orizzontali. A entrambe le funzioni vengono passate le variabili contenenti handle dell'immagine, larghezza, altezza, colore di primo piano e bordo. Queste sono tutte le modifiche che occorre apportare alla funzione `graph()`. Si devono però ancora costruire due

funzioni: `drawAxis()` e `drawDashedLines()`. La prima è la più semplice e utilizza i propri parametri per disegnare l'asse del grafico come un rettangolo:

```
function drawAxis($image, $w, $h, $col, $top,$right,$left,$bottom) {
   ImageRectangle($image,$left,$top,$w-$right,$h-$bottom,$col);
}
```

La funzione `drawDashedLines()` è un po' più complessa. Per prima cosa calcola l'altezza del grafico, tenendo conto dei bordi superiore e inferiore. Quindi calcola la spaziatura tra le righe tratteggiate, il cui valore è memorizzato nella variabile `$space`. Infine utilizza un ciclo `for` per visualizzare le quattro righe nel grafico:

```
function drawLines($image, $w, $h, $col, $top,$right,$left,$bottom) {
   $gHeight = $h - ($top + $bottom);
   $space = $gHeight / 4;
   $y = $top;
   for($a=1;$a<5;$a++) {
      ImageLine($image,$left,$y,$w-$right,$y,$col);
      $y=$y+$space;
   }
}
```

L'invocazione della funzione `graph()` deve contenere un parametro aggiuntivo che specifica il colore di primo piano, che in questo esempio è azzurro chiaro:

```
graph("graphics/graph.jpeg", 200, 150, 14, 2);
```

Lo script sviluppato sinora è il seguente:

```
<?php

// Grafico GD - Esempio 31-2
//--------------------------------

function graph($name,$w,$h,$bgc,$fgc) {
   $top=15;
   $right=5;
   $left=35;
   $bottom=25;
   $image = ImageCreate($w,$h);
   $cols = colours($image);
   ImageFill($image,0,0,$cols[$bgc]);
   drawAxis($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
   drawLines($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
   ImageJPEG($image, $name);
   ImageDestroy($image);
}

function colours($image) {
   $white = ImageColorAllocate($image,255,255,255);
   $black = ImageColorAllocate($image,0,0,0);
   $lightblue =  ImageColorAllocate($image,64,64,255);
   $blue = ImageColorAllocate($image,0,0,255);
   $darkblue = ImageColorAllocate($image,0,0,192);
   $red = ImageColorAllocate($image,255,0,0);
   $lightred = ImageColorAllocate($image,255,64,64);
```

```
    $darkred = ImageColorAllocate($image,192,0,0);
    $green = ImageColorAllocate($image,0,255,0);
    $lightgreen = ImageColorAllocate($image,64,255,64);
    $darkgreen = ImageColorAllocate($image,0,192,0);
    $yellow = ImageColorAllocate($image,255,255,0);
    $grey = ImageColorAllocate($image,192,192,192);
    $darkgrey = ImageColorAllocate($image,128,128,128);
    $lightgrey = ImageColorAllocate($image,192,192,255);
    $cols = array($white,$black,$lightblue,$blue,$darkblue,$red,
              $lightred,$darkred,$green, $lightgreen,$darkgreen,
              $yellow,$grey,$darkgrey,$lightgrey);
    return $cols;
}


function drawAxis($image, $w, $h, $col, $top,$right,$left,$bottom) {
    ImageRectangle($image,$left,$top,$w-$right,$h-$bottom,$col);
}


function drawLines($image, $w, $h, $col, $top,$right,$left,$bottom) {
    $gHeight = $h - ($top + $bottom);
    $space = $gHeight / 4;
    $y = $top;
    for($a=1;$a<5;$a++) {
        ImageLine($image,$left,$y,$w-$right,$y,$col);
        $y=$y+$space;
    }
}


graph("graphics/graph.jpeg", 200, 150, 14, 2);

?>
<img src='graphics/graph.jpeg'>
```

L'output ottenuto dalla funzione `graph()` è quello della Figura 31.5.

Creazione del titolo e delle etichette per gli assi

La fase successiva riguarda l'inserimento del titolo e delle etichette per gli assi X e Y. La funzione `graph()` viene modificata in modo da includere altri tre parametri: il titolo e le due etichette.

```
function graph($name,$w,$h,$bgc,$fgc,$title,$xtitle,$ytitle) {
```

Successivamente, viene inclusa una chiamata a funzione dopo l'invocazione di `drawLines()`, che visualizza queste etichette:

```
drawTitles($image,$w,$h,$cols[$fgc],$title,$xtitle,$ytitle);
```

A questo punto è necessario includere anche la funzione `drawTitles()`, che calcola la lunghezza del titolo e delle due etichette e le visualizza al centro, in posizioni

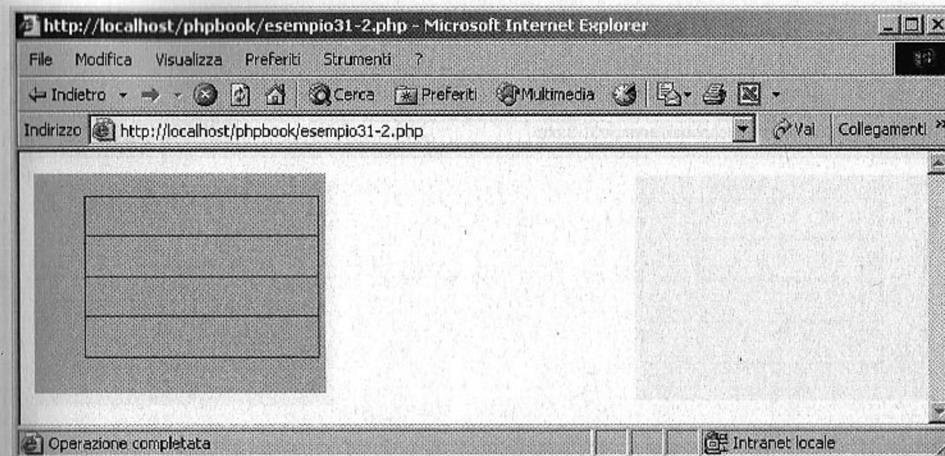


Figura 31.5
Grafico parte 2, bordi e righelli orizzontali.

fisse dell'immagine. Non viene eseguito alcun test per verificare se il testo si adatta alla posizione. Si presuppone che lo sviluppatore si accerti prima che il grafico sia sufficientemente grande per visualizzare i titoli e le etichette:

```
function drawTitles($image,$w,$h,$col,$title,$xtitle,$ytitle) {
    $width = ImageFontWidth(1) * strlen($title);
    $x = ($w - $width)/2;
    ImageString($image,1,$x,3,$title,$col);
    $width = ImageFontWidth(1) * strlen($ytitle);
    $x = ($w - $width)/2;
    ImageString($image,1,$x,$h-10,$ytitle,$col);
    $height = ImageFontWidth(1) * strlen($xtitle);
    $y = ($h - $height)/2;
    ImageStringUp($image,1,1,$y+$height,$xtitle,$col);
}
```

Infine viene modificata la chiamata alla funzione `graph()` per includere gli altri tre parametri:

```
graph("graphics/graph.jpeg", 200, 150, 14,2,"Totale delle vendite",
      "Vendite (euro)","Giorni feriali");
```

L'output ottenuto dalla funzione `graph()` è quello della Figura 31.6.

Creazione delle etichette dei dati per l'asse X

Per creare le etichette dell'asse X dovete aggiungere due parametri nuovi alla funzione `graph()`, ossia il numero delle etichette dei dati (e di dati che arriveranno) e un array contenente le etichette:

```
function graph($name,$w,$h,$bgc,$fgc,$title,$xtitle,$ytitle,$numX,
$xTitles) {
```

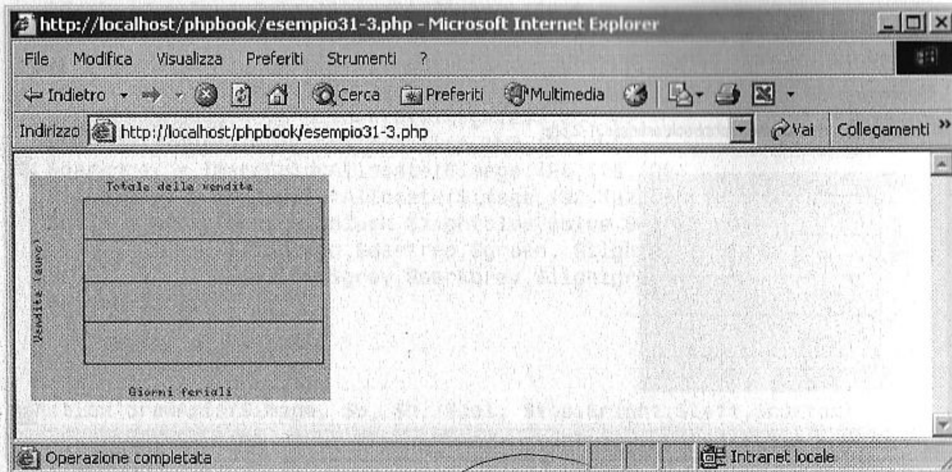


Figura 31.6
Grafico parte 3, titolo ed etichette.

Dopo l'invocazione della funzione `drawTitles()` viene inserita una chiamata a funzione che visualizza le seguenti etichette:

```
drawXLables($image,$w,$h,$right,$left,$cols[$fgc],$numX,$xTitles);
```

Occorre anche includere la funzione `drawXLables()`, che calcola la spaziatura tra ciascuna etichetta e poi, con un ciclo `for`, le visualizza nel grafico:

```
function drawXLables($image,$w,$h,$right,$left,$col,$numX,$xTitles) {
   $graphWidth = $w - ($right+$left);
   $spacing = $graphWidth/$numX;
   $x = $left + ($spacing/4);
   for($a=0;$a<$numX;$a++) {
      ImageString($image,1,$x,$h-20,$xTitles[$a],$col);
      $x=$x+$spacing;
   }
}
```

Ancora una volta è compito dello sviluppatore verificare che ci sia spazio sufficiente per ospitare le etichette nel grafico. Infine, prima di invocare la funzione `graph()`, è necessario definire un array che contenga le etichette dell'asse X. La chiamata alla funzione `graph()` dev'essere modificata per contenere il numero di etichette e l'array:

```
$xTitles = array("Lun","Mar","Mer","Gio","Ven");
graph("graphics/graph.jpeg", 200, 150, 14,2,"Totale delle
vendite","Vendite (euro)","Giorni feriali",5,$xTitles);
```

L'output ottenuto dalla funzione `graph()` è quello della Figura 31.7.

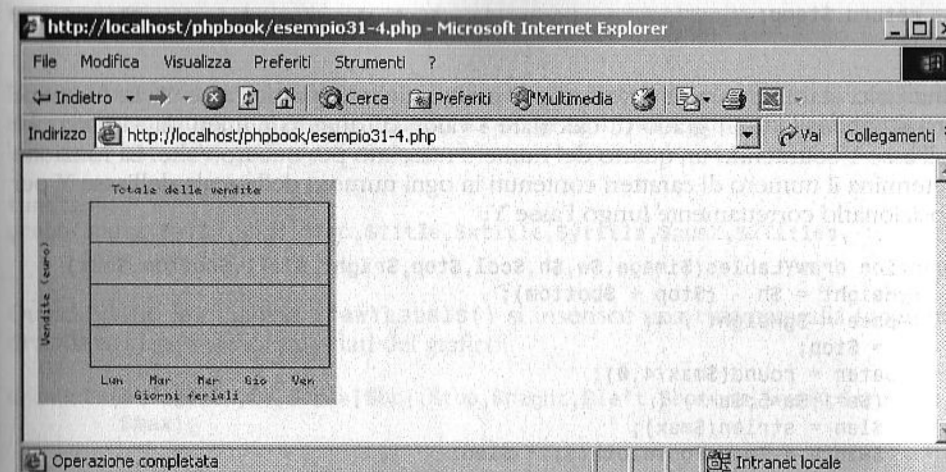


Figura 31.7
Grafico parte 4, etichette per l'asse X.

Creazione della scala numerica per l'asse Y

Per visualizzare la scala numerica dell'asse Y, dovete sapere quali valori compar ranno nella scala. Pertanto dovete modificare la funzione `graph()` in modo che valori possano essere passati come un array e memorizzati nella variabile `$xVa lues`:

```
function graph($name,$w,$h,$bgc,$fgc,$title,$xtitle,$ytitle,$numX,
$xTitles,$xValues) {
```

Successivamente, si ricorre a una chiamata alla funzione `calcMaxDataItem()` pe stabilire quale sia il valore più grande, in modo da calcolare la scala per l'asse Y Questa chiamata viene inserita dopo quella alla funzione `drawXLables()`:

```
$max = calcMaxDataItem($xValues,$numX);
```

Con una chiamata alla funzione `drawYLables()` potrete visualizzare i valori nume rici:

```
drawYLables($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom,$max);
```

La funzione `calcMaxDataItem()` utilizza un semplice ciclo `for` per esaminare ci scun dato e stabilire quale sia il più grande:

```
function calcMaxDataItem($xValues, $numX) {
   $temp=0;
   for($a=0;$a<$numX;$a++) {
      if($xValues[$a] > $temp)
         $temp = $xValues[$a];
   }
```

```
    return $temp;
}
```

Una volta stabilito qual è il valore più grande della scala dell'asse Y, la funzione `drawYLables()` è in grado di calcolare i valori di quattro numeri da visualizzare sull'asse Y sottraendo un quarto del numero massimo per quattro volte. La funzione determina il numero di caratteri contenuti in ogni numero della scala dell'asse Y per posizionarlo correttamente lungo l'asse Y:

```
function drawYLables($image,$w,$h,$col,$top,$right,$left,$bottom,$max) {
    $gHeight = $h - ($top + $bottom);
    $space = $gHeight / 4;
    $y = $top;
    $quater = round($max/4,0);
    for($a=1;$a<5;$a++) {
        $len = strlen($max);
        $width = ImageFontWidth(1) * $len;
        ImageString($image,0,32-$width,$y-2,$max,$col);
        $y=$y+$space;
        $max=$max-$quater;
    }
}
```

Infine viene creato un array per memorizzare i valori e modificare la chiamata alla funzione `graph()` per passarle il seguente array:

```
$xTitles = array("Lun","Mar","Mer","Gio","Ven");
$xValues = array(50,55,78,87,32);
graph("graphics/graph.jpeg", 200, 150, 14,2,"Totale delle vendite",
    "Vendite (euro)","Giorni feriali",5,$xTitles,$xValues,5);
```

L'output ottenuto dalla funzione `graph()` è quello della Figura 31.8.

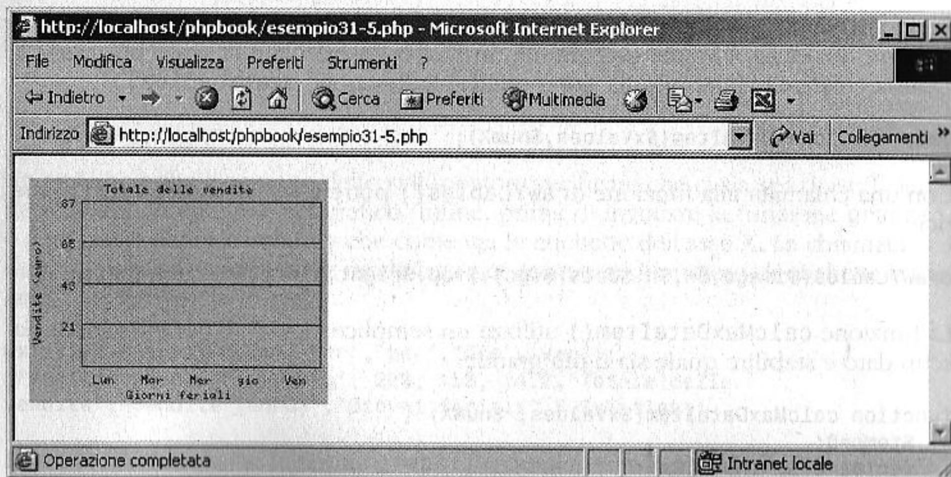


Figura 31.8
Grafico parte 5, numeri dell'asse Y.

Visualizzazione dei dati

L'ultima operazione consiste nella visualizzazione dei dati nel grafico. Si comincia inserendo un ultimo parametro nella funzione `graph()` che conterrà il valore di colore delle barre del grafico:

```
function
graph($name,$w,$h,$bgc,$fgc,$title,$xtitle,$ytitle,$numX,$xTitles,
    $xValues,$bc) {
```

Quindi, dopo la funzione `drawYLabels()` si inserisce una chiamata alla funzione `drawData()` per elaborare i dati del grafico:

```
drawData($image,$w,$h,$cols[$bc],$top,$right,$left,$bottom,$numX,$xValues,
    $max);
```

A questo punto si utilizza la funzione `drawData()`, che calcola la dimensione di ogni barra del grafico basandosi sulla dimensione complessiva dell'immagine e sul valore dei dati. Ogni barra viene visualizzata con la funzione `ImageFilledRectangle()`:

```
function
drawData($image,$w,$h,$col,$top,$right,$left,$bottom,$numX,$xValues,$max)
{
    $graphHeight = $h - ($top+$bottom);
    $pixelValue = $graphHeight / $max;
    $graphWidth = $w - ($right+$left);
    $spacing = $graphWidth/$numX;
    $gap = $spacing/4;
    $x = $left + ($spacing/6);
    for($a=0;$a<$numX;$a++) {
        $rectSize = ($graphHeight + $top) - $xValues[$a]*$pixelValue;
        ImageFilledRectangle($image,$x,$rectSize,$x+($spacing-$gap),
                $h-$bottom,$col);
        $x=$x+$spacing;
    }
}
```

Infine, si modificha l'invocazione della funzione `graph()` per includere un parametro che rappresenta il colore delle barre, in questo esempio rosso:

```
$xTitles = array("Lun","Mar","Mer","Gio","Ven");
$xValues = array(50,55,78,87,32);
graph("graphics/graph.jpeg", 200, 150, 14,2,"Totale delle vendite",
    "Vendite (euro)","Giorni feriali",5,$xTitles,$xValues,5);
```

Lo script conclusivo è il seguente:

```
<?php

// Grafico GD - Esempio 31-6
//-----------------------------

function
graph($name,$w,$h,$bgc,$fgc,$title,$xtitle,$ytitle,$numX,$xTitles,$xValues
, $bc) {
```

```
    $top=15;
    $right=5;
    $left=35;
    $bottom=25;
    $image = ImageCreate($w,$h);
    $cols = colours($image);
    ImageFill($image,0,0,$cols[$bgc]);
    drawAxis($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
    drawLines($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom);
    drawTitles($image,$w,$h,$cols[$fgc],$title,$xtitle,$ytitle);
    drawXLables($image,$w,$h,$right,$left,$cols[$fgc],$numX,$xTitles);
    $max = calcMaxDataItem($xValues,$numX);
    drawYLables($image,$w,$h,$cols[$fgc],$top,$right,$left,$bottom,$max);
    drawData($image,$w,$h,$cols[$bc],$top,$right,$left,$bottom,$numX,
             $xValues,$max);
    ImageJPEG($image, $name);
    ImageDestroy($image);
}

function colours($image) {
    $white = ImageColorAllocate($image,255,255,255);
    $black = ImageColorAllocate($image,0,0,0);
    $lightblue =  ImageColorAllocate($image,64,64,255);
    $blue = ImageColorAllocate($image,0,0,255);
    $darkblue = ImageColorAllocate($image,0,0,192);
    $red = ImageColorAllocate($image,255,0,0);
    $lightred = ImageColorAllocate($image,255,64,64);
    $darkred = ImageColorAllocate($image,192,0,0);
    $green = ImageColorAllocate($image,0,255,0);
    $lightgreen = ImageColorAllocate($image,64,255,64);
    $darkgreen = ImageColorAllocate($image,0,192,0);
    $yellow = ImageColorAllocate($image,255,255,0);
    $grey = ImageColorAllocate($image,192,192,192);
    $darkgrey = ImageColorAllocate($image,128,128,128);
    $lightgrey = ImageColorAllocate($image,192,192,255);
    $cols = array($white,$black,$lightblue,$blue,$darkblue,$red,
                  $lightred,$darkred,$green,$lightgreen,$darkgreen,
                  $yellow,$grey,$darkgrey,$lightgrey);
    return $cols;
}

function drawAxis($image, $w, $h, $col, $top,$right,$left,$bottom) {
    ImageRectangle($image,$left,$top,$w-$right,$h-$bottom,$col);
}

function drawLines($image, $w, $h, $col, $top,$right,$left,$bottom) {
    $gHeight = $h - ($top + $bottom);
    $space = $gHeight / 4;
    $y = $top;
    for($a=1;$a<5;$a++) {
       ImageLine($image,$left,$y,$w-$right,$y,$col);
       $y=$y+$space;
    }
}

function drawTitles($image,$w,$h,$col,$title,$xtitle,$ytitle) {
    $width = ImageFontWidth(1) * strlen($title);
```



```
    $x = ($w - $width)/2;
    ImageString($image,1,$x,3,$title,$col);
    $width = ImageFontWidth(1) * strlen($ytitle);
    $x = ($w - $width)/2;
    ImageString($image,1,$x,$h-10,$ytitle,$col);
    $height = ImageFontWidth(1) * strlen($xtitle);
    $y = ($h - $height)/2;
    ImageStringUp($image,1,1,$y+$height,$xtitle,$col);
}

function drawXLables($image,$w,$h,$right,$left,$col,$numX,$xTitles) {
    $graphWidth = $w - ($right+$left);
    $spacing = $graphWidth/$numX;
    $x = $left + ($spacing/4);
    for($a=0;$a<$numX;$a++) {
       ImageString($image,1,$x,$h-20,$xTitles[$a],$col);
       $x=$x+$spacing;
    }
}

function calcMaxDataItem($xValues, $numX) {
    $temp=0;
    for($a=0;$a<$numX;$a++) {
       if($xValues[$a] > $temp)
          $temp = $xValues[$a];
    }
    return $temp;
}

function drawYLables($image,$w,$h,$col,$top,$right,$left,$bottom,$max) {
    $gHeight = $h - ($top + $bottom);
    $space = $gHeight / 4;
    $y = $top;
    $quater = round($max/4,0);
    for($a=1;$a<5;$a++) {
       $len = strlen($max);
       $width = ImageFontWidth(1) * $len;
       ImageString($image,0,32-$width,$y-2,$max,$col);
       $y=$y+$space;
       $max=$max-$quater;
    }
}

function drawData($image,$w,$h,$col,$top,$right,$left,$bottom,$numX,
$xValues,$max) {
    $graphHeight = $h - ($top+$bottom);
    $pixelValue = $graphHeight / $max;
    $graphWidth = $w - ($right+$left);
    $spacing = $graphWidth/$numX;
    $gap = $spacing/4;
    $x = $left + ($spacing/6);
    for($a=0;$a<$numX;$a++) {
       $rectSize = ($graphHeight + $top) - $xValues[$a]*$pixelValue;
       ImageFilledRectangle($image,$x,$rectSize,$x+($spacing-$gap),
                            $h-$bottom,$col);
       $x=$x+$spacing;
```

```
    }
}

$xTitles = array("Lun","Mar","Mer","Gio","Ven");
$xValues = array(50,55,78,87,32);
graph("graphics/graph.jpeg", 200, 150, 14,2,"Totale delle vendite",
      "Vendite (euro)","Giorni feriali",5,$xTitles,$xValues,5);

?>
<img src='graphics/graph.jpeg'>
```

L'output ottenuto dalla funzione `graph()` è quello della Figura 31.9.

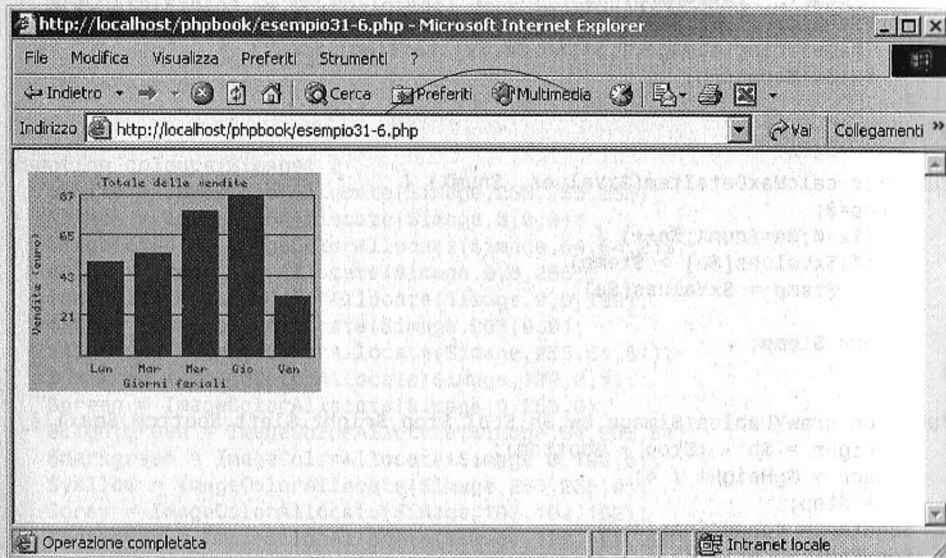


Figura 31.9
Grafico parte 6, grafico ultimato.

Ecco fatto. Sono stati effettuati sei passaggi per completare un'immagine utile che può essere utilizzata per molte pagine Web.

Riepilogo

Questo capitolo ha illustrato come utilizzare la libreria GD per creare un'immagine utile e sufficientemente flessibile per essere utilizzata in una vasta gamma di situazioni. Il prossimo capitolo introduce la libreria di funzioni PDFLib e spiega come utilizzare queste funzioni per creare documenti PDF personalizzati.

Parte IX

La libreria PDF

32 Creazione di documenti PDF

Capitolo 32

Creazione di documenti PDF

Introduzione

PDFlib è una libreria di funzioni creata per consentire la generazione di documenti nel formato PDF (*Portable Document Format*) di Adobe. Il formato PDF è diventato uno standard per l'archiviazione di documenti sul Web.
La libreria PDFlib ora è disponibile per PHP e le versioni più recenti di PHP la includono già nel pacchetto. Quindi dovete soltanto attivarla per poter iniziare a dar vita a documenti PDF dinamici.
Questo capitolo offre un'introduzione alla libreria PDFLib e analizza alcune funzioni a disposizione degli sviluppatori.

Attivazione della libreria PDFlib

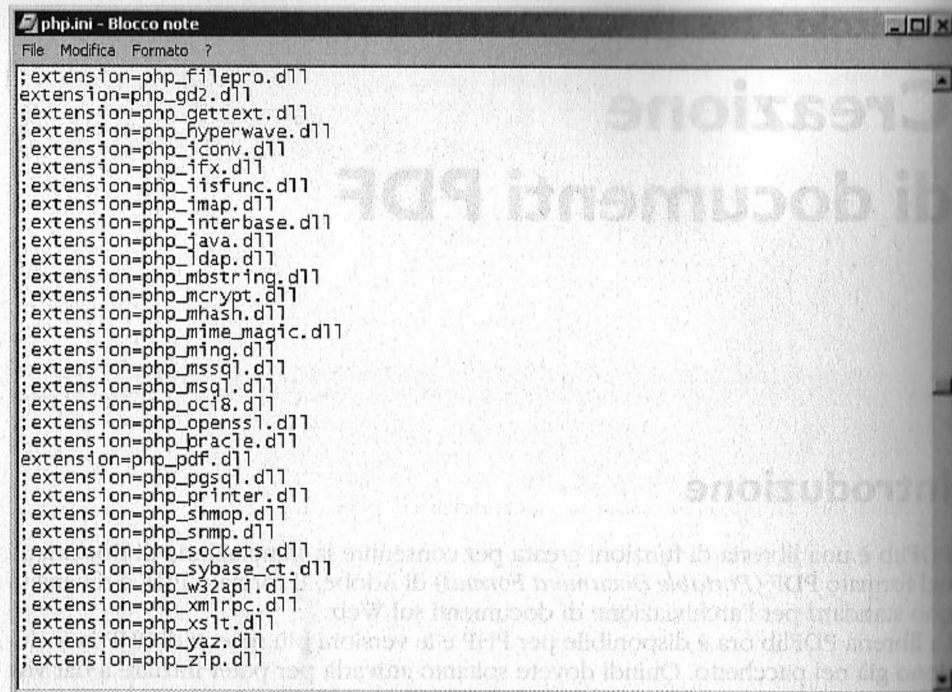
La libreria di funzioni PDFlib viene fornita con l'installazione di PHP, tuttavia è necessario modificare il file php.ini per attivarla. Questo file si trova nella directory *c:\windows* o *c:\winnt* di un sistema Windows. Per attivare la libreria PDFlib dovete aprire il file php.ini in un editor come Blocco note e poi trovare la riga seguente nel file:

```
;extension=php_pdf.dll
```

Per attivare la libreria è sufficiente rimuovere il ; dall'inizio della riga e salvare il file. La Figura 32.1 illustra il file php.ini visualizzato in Blocco note. Ora siete pronti per iniziare a sviluppare con PDFlib. Per visualizzare l'output creato dagli script di questo capitolo dovete procurarvi una copia di Adobe® Acrobat® Reader, disponibile presso `www.adobe.it`.

Perché utilizzare PDFlib

La libreria PDFlib è stata sviluppata per consentire la creazione di documenti PDF dinamici tramite il World Wide Web. È possibile creare un documento PDF che include dati ricevuti da un utente Web e/o provenienti da un database con capacità


php.ini - Blocco note

File Modifica Formato ?

```
;extension=php_filepro.dll
extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_hyperwave.dll
;extension=php_iconv.dll
;extension=php_ifx.dll
;extension=php_iisfunc.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_java.dll
;extension=php_ldap.dll
;extension=php_mbstring.dll
;extension=php_mcrypt.dll
;extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_mssql.dll
;extension=php_msql.dll
;extension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_oracle.dll
extension=php_pdf.dll
;extension=php_pgsql.dll
;extension=php_printer.dll
;extension=php_shmop.dll
;extension=php_snmp.dll
;extension=php_sockets.dll
;extension=php_sybase_ct.dll
;extension=php_w32api.dll
;extension=php_xmlrpc.dll
;extension=php_xslt.dll
;extension=php_yaz.dll
;extension=php_zip.dll
```

Figura 32.1
Il file php.ini.

Web, il che significa ché è possibile dar vita a documenti PDF personalizzati in modo dinamico.
PDFlib si integra senza problemi in PHP, eliminando la necessità di seguire il tradizionale percorso di creazione che dal postscript passa ad Acrobat Distiller e poi al PDF.
Inoltre è molto rapida e, quindi, ideale per la generazione di documenti dinamici, ed è disponibile per diverse piattaforme, nonché per altri linguaggi di programmazione oltre a PHP. La documentazione su PDFlib è disponibile al sito: `www.pdflib.com`.

Creazione del primo documento PDF

Per creare un documento PDF (anche il più semplice), è necessario seguire cinque passaggi principali e introdurre diverse nuove funzioni. La prima cosa da fare è indicare che si desidera iniziare a creare un documento PDF; ciò è possibile con la funzione `pdf_new()`.
La sintassi della funzione è la seguente.

```
int pdf_new(void);
```


La tabella seguente descrive il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Restituzione di `pdf_new()`	`int`	La funzione restituisce un handle a un documento PDF. Questo handle sarà utilizzato in tutte le funzioni successive che riguardano questo particolare documento PDF.

Esempio di funzione:

```
$myPDF = pdf_new();
```

Un volta ottenuto un riferimento (handle) per un documento PDF, è necessario assegnargli un nome di file che sarà utilizzato per memorizzare il file PDF sul server. A tal fine si utilizza la funzione `pdf_open_file()`.
La sintassi della funzione è la seguente.

```
int pdf_open_file(int pdfHandle, string nomefile)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
nomefile	`string`	Nome del file di documento PDF che viene creato.
Restituzione di `pdf_open()`	`int`	Restituisce `TRUE` se la funzione ha esito positivo o `FALSE` in caso contrario.

Esempio di funzione:

```
pdf_open_file($myPDF,"primo.pdf");
```

Finora è stato creato un documento PDF che, però, al momento non contiene alcunché. Il prossimo passaggio consiste nell'inserire una pagina nel documento con la funzione `pdf_begin_page()`.
La sintassi della funzione è la seguente.

```
void pdf_begin_page(int pdfHandle, double larghezza, double altezza);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
larghezza	`double`	Larghezza della pagina specificata in punti.
altezza	`double`	Altezza della pagina specificata in punti.
Restituzione di `pdf_begin_page()`	`void`	Non restituisce alcunché.

Esempio di funzione:

```
pdf_begin_page($myPDF,595,842);
```

La larghezza e l'altezza di una pagina sono specificate in punti, dove 1 punto equivale a 1/72 di pollice. Per fortuna, normalmente si creano pagine di dimensioni prestabilite, per esempio A4. La Tabella 32.1 fornisce la larghezza e l'altezza in punti dei formati di pagina più comuni:

Tabella 32.1 Dimensioni delle pagine

Dimensione pagina	Larghezza	Altezza
A0	2380	3368
A1	1684	2380
A2	1190	1684
A3	842	1190
A4	595	842
A5	421	595
A6	297	421
B5	501	709
Letter	612	792
Legal	612	1008

Una volta creata la pagina, è possibile chiuderla con la funzione `pdf_end_page()` (un segnale che si è finito di lavorare con essa).
La sintassi della funzione è la seguente.

```
Void pdf_end_page(int pdfHandle)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
Restituzione di `pdf_end_page()`	void	Non restituisce alcunché.

Esempio di funzione:

```
pdf_end_page($myPDF);
```

Una serie di chiamate ripetute a `pdf_begin_page()` e `pdf_end_page()` consente di dar vita a più di una pagina. L'ultimo passo nella creazione del documento è la chiamata alla funzione `pdf_close()`.
La sintassi della funzione è la seguente.

```
void pdf_end_page(int pdfHandle)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
Restituzione di `pdf_close()`	void	Non restituisce alcunché.

La combinazione di queste funzioni in uno script PHP produce quanto segue:

```
<?php

// Documenti PDF - Esempio 32-1
//---------------------------------

$pageWidth=595;
$pageHeight=842;

$myPDF = pdf_new();
pdf_open_file($myPDF,"pdf/primo.pdf");
pdf_begin_page($myPDF,$pageWidth,$pageHeight);

pdf_end_page($myPDF);
pdf_close($myPDF);

?>
<a href="pdf/primo.pdf">Visualizza il documento PDF generato.</a>
```

Questo script include un collegamento ipertestuale che consente di visualizzare il file PDF nel browser Web. Se fate clic sul collegamento, Adobe Acrobat Reader verrà avviato e potrete vedere il documento appena generato. Sfortunatamente vedrete che questo documento PDF è costituito da una sola pagina vuota. Occorre pertanto introdurre alcune altre funzioni per poter avere documenti più interessanti.

Inserimento di testo

Per migliorare il documento PDF appena creato si può aggiungere del testo. A tal fine si farà ricorso a tre funzioni: `pdf_findfont()`, `pdf_setfont()` e `pdf_show_xy()`. La prima di queste, `pdf_findfont()`, ha la sintassi seguente.

```
int pdf_findfont(int pdfHandle, string nomeFont, string codifica, int
incorporato);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
nomeFont	string	Il nome del font da utilizzare. Tra i font più comuni: "Times New Roman", "Arial" e "Courier".
codifica	string	Seleziona la particolare codifica dei caratteri per il font selezionato. La codifica è un codice numerico a 8 bit mappato con un carattere particolare nel set del font. Normalmente si utilizza il valore "host" per la stringa di codifica, in quanto seleziona il sistema di codifica appropriato per il sistema operativo del server.
incorporato	int	Uno switch binario, dove 1 indica che il font sarà incorporato nel documento PDF e 0 indica che non lo sarà.
Restituzione di `pdf_findfont()`	int	Restituisce un handle per il font selezionato. Di seguito trovate un esempio di questa funzione.

Esempio di funzione:

```
$arial = pdf_findfont($myPDF, "Arial", "host", 1);
```

Questa istruzione creerà un font Arial che viene incorporato nel documento PDF. Una volta definito un font, si può scegliere di utilizzarlo con la funzione `pdf_setfont()`.
La sintassi della funzione è la seguente.

```
void pdf_setfont(int pdfHandle, int font, double dimensioneFont)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato
font	int	L'handle del font restituito dalla funzione `pdf_find_font()`.
dimensioneFont	double	La dimensione del font in punti.
Restituzione di `pdf_setfont()`	void	Non restituisce alcunché.

Esempio di funzione:

```
pdf_setfont($myPDF, $arial, 12);
```

Questa istruzione imposta il font Arial specificato in precedenza a una dimensione di 12 punti. Ora è possibile scrivere del testo in un certa posizione della pagina. A tal fine si utilizza la funzione `pdf_show_xy`.
La sintassi della funzione è la seguente.

```
void pdf_show_xy(int pdfHandle, string testo, double x, double y)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato
testo	string	Il testo da visualizzare.
x	double	La coordinata X sulla pagina.
y	double	La coordinata Y sulla pagina.
Restituzione di `pdf_show_xy()`	void	Non restituisce alcunché.

Esempio di funzione:

```
pdf_show_xy($myPDF, "Benvenuti nel mondo del PDF",70,700);
```

Incorporando queste funzioni nello script precedente si ottiene quanto segue:

```
<?php

// Documenti PDF - Esempio 32-2
//---------------------------------
```



```
$pageWidth=595;
$pageHeight=842;

$myPDF = pdf_new();
pdf_open_file($myPDF,"pdf/secondo.pdf");
pdf_begin_page($myPDF,$pageWidth,$pageHeight);

$arial = pdf_findfont($myPDF, "Arial", "host", 1);
pdf_setfont($myPDF, $arial, 12);
pdf_show_xy($myPDF, "Benvenuti nel mondo del PDF", 70,700);

pdf_end_page($myPDF);
pdf_close($myPDF);

?>
<a href="pdf/secondo.pdf">Visualizza il documento PDF generato.</a>
```

L'output di questo script è quello della Figura 32.2.

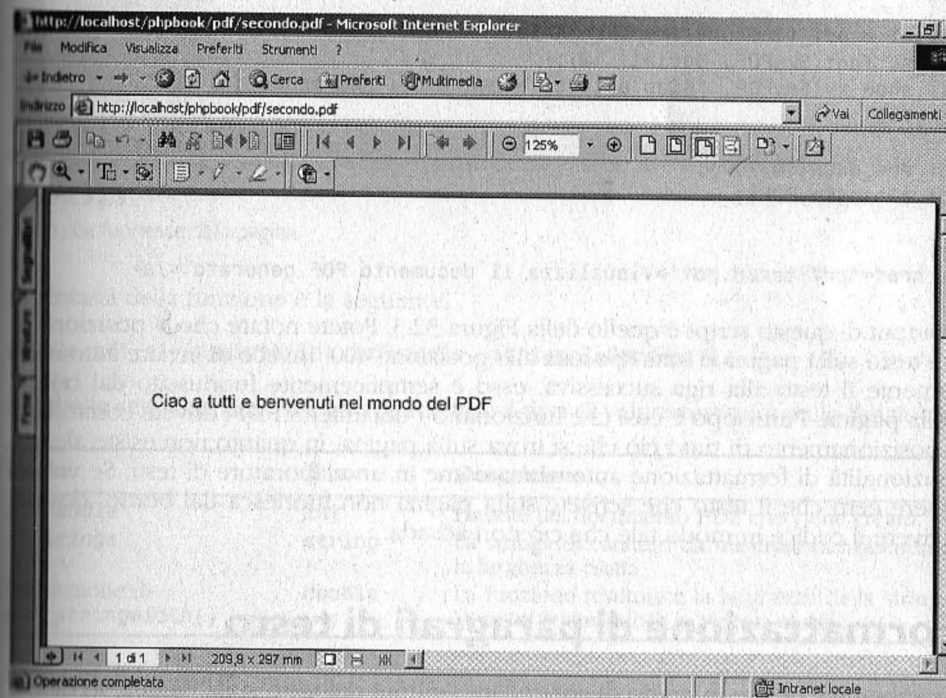


Figura 32.2
Output di testo in PDF.

La Figura 32.2 mostra che il testo "Benvenuti nel mondo del PDF" è stato visualizzato alla posizione 70 per 700 sulla pagina.

Righe di testo che fuoriescono dalla pagina

Finora sembra andare tutto bene. Nell'esempio precedente siete riusciti a creare un documento PDF che contiene testo. Sfortunatamente, però, le cose non sono così semplici. Lo script che segue modifica l'esempio precedente allungando un po' il testo e spostandone la posizione di inizio.

```
<?php

// Documenti PDF - Esempio 32-3
//---------------------------------

$pageWidth=595;
$pageHeight=842;

$myPDF = pdf_new();
pdf_open_file($myPDF,"pdf/terzo.pdf");
pdf_begin_page($myPDF,$pageWidth,$pageHeight);

$arial = pdf_findfont($myPDF, "Arial", "host", 1);
pdf_setfont($myPDF, $arial, 12);
pdf_show_xy($myPDF, "Ciao a tutti e benvenuti nel mondo del PDF",
400,700);

pdf_end_page($myPDF);
pdf_close($myPDF);

?>
<a href="pdf/terzo.pdf">Visualizza il documento PDF generato.</a>
```

L'output di questo script è quello della Figura 32.3. Potete notare che la posizione X del testo sulla pagina è stata spostata alla posizione 400. Invece di inviare automaticamente il testo alla riga successiva, esso è semplicemente fuoriuscito dal bordo della pagina. Purtroppo è così che funzionano i documenti PDF: dovete controllare il posizionamento di tutto ciò che si trova sulla pagina, in quanto non esiste alcuna funzionalità di formattazione automatica come in un elaboratore di testi. Se volete essere certi che il testo che scrivete sulla pagina non fuoriesca dal bordo, dovete scrivere il codice in modo tale che ciò non accada.

Formattazione di paragrafi di testo

Sarebbe utile essere in grado di controllare l'output del testo sulla pagina, in modo che tutto sia al suo posto e nessuna parte del testo scompaia fuori dal bordo. A tal fine dovreste costruire una riga di testo parola per parola, controllando che ciascuna parola aggiunta alla frase sia contenuta nella riga. Quando si raggiunge il limite, la riga di parole dev'essere aggiunta alla pagina e occorre iniziare un'altra riga. Questa operazione dovrebbe essere ripetuta fino a quando tutte le parole sono state inserite nel file. Ovviamente, per poter agire in questo modo dovrebbe essere possibile calcolare la larghezza esatta di una parola in punti. Fortunatamente esiste una funzione chiamata `pdf_stringwidth()` che si occupa di questo calcolo.

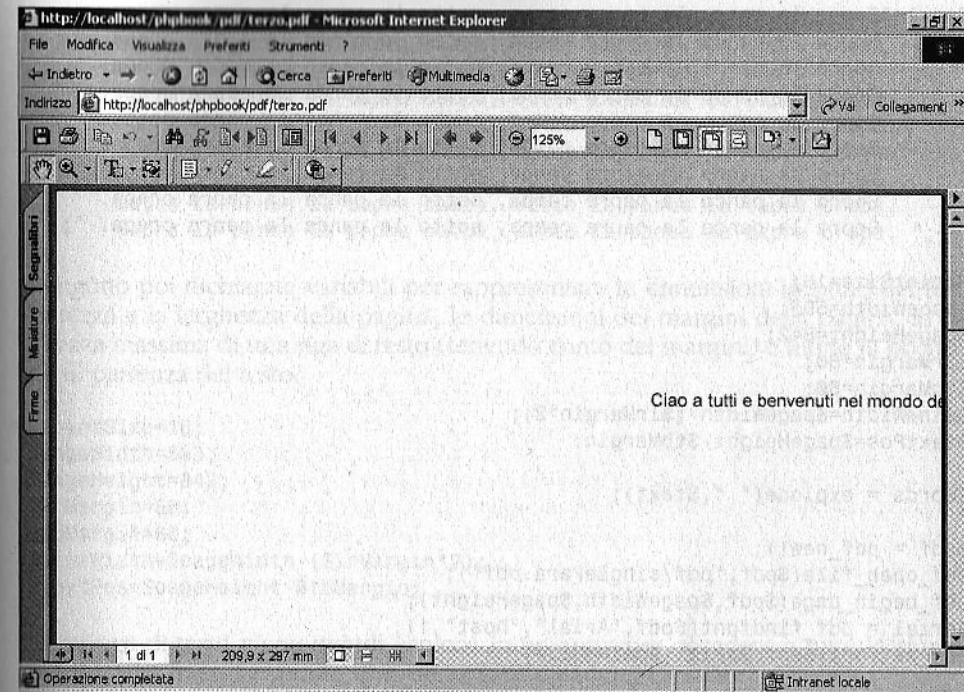


Figura 32.3
Testo che fuoriesce dalla pagina.

La sintassi della funzione è la seguente.

```
double pdf_stringwidth(int pdfHandle, string laStringa);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
laStringa	string	La stringa di caratteri da misurare per ottenerne la larghezza esatta.
Restituzione di `pdf_stringwidth()`	double	La funzione restituisce la larghezza della stringa in punti come valore in virgola mobile.

Esempio di funzione:

```
pdf_stringwidth($pdf,$line);
```

Grazie a questa funzione è possibile essere certi che il testo sarà contenuto nella pagina. Lo script che segue illustra il metodo per la formattazione di un paragrafo di testo.

```
<?php

// Documenti PDF - Esempio 32-4
//------------------------------
```

```
$text="Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa. ";

$pointSize=16;
$pageWidth=595;
$pageHeight=842;
$lrMargin=80;
$tbMargin=80;
$lineWidth=$pageWidth-($lrMargin*2);
$textPos=$pageHeight-$tbMargin;

$words = explode(" ",$text);

$pdf = pdf_new();
pdf_open_file($pdf,"pdf/singlePara.pdf");
pdf_begin_page($pdf,$pageWidth,$pageHeight);
$arial = pdf_findfont($pdf,"Arial","host",1);
pdf_setfont($pdf,$arial,$pointSize);

$lines = array("");
$line=0;
foreach($words as $word){
   if(pdf_stringwidth($pdf,$lines[$line] . " " . $word) < $lineWidth){
      $lines[$line] = $lines[$line] . $word . " ";
   }
   else {
      $line++;
      $lines[$line] = $word . " ";
   }
}

pdf_set_text_pos($pdf,$lrMargin,$pageHeight-$tbMargin);

foreach($lines as $aline) {
   pdf_show_xy($pdf,$aline,$lrMargin,$textPos);
   $textPos=$textPos-($pointSize+2);
}

pdf_end_page($pdf);
pdf_close($pdf);

?>
<a href="pdf/SinglePara.pdf">Visualizza il documento PDF generato.</a>
```

Lo script inizia dichiarando la stringa di testo che dovete visualizzare sulla pagina PDF:

```
<?php

// Documenti PDF - Esempio 32-4
//-----------------------------
```



```
$text="Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa.
      Sopra la panca la capra campa, sotto la panca la capra crepa. ";
```

Vengono poi dichiarate variabili per rappresentare le dimensioni in punti del font, l'altezza e la larghezza della pagina, le dimensioni dei margini della pagina, la larghezza massima di una riga di testo (tenendo conto dei margini) e infine la posizione di partenza del testo.

```
$pointSize=16;
$pageWidth=595;
$pageHeight=842;
$lrMargin=80;
$tbMargin=80;
$lineWidth=$pageWidth-($lrMargin*2);
$textPos=$pageHeight-$tbMargin;
```

La stringa di testo viene quindi esplosa in un array per accedere a tutte le singole parole della frase. Le parole sono separate dal carattere spazio:

```
$words = explode(" ",$text);
```

La parte successiva dello script dichiara un nuovo documento PDF, gli dà un nome, inserisce una pagina e infine seleziona e assegna un font da utilizzare:

```
$pdf = pdf_new();
pdf_open_file($pdf,"pdf/singlePara.pdf");
pdf_begin_page($pdf,$pageWidth,$pageHeight);
$arial = pdf_findfont($pdf,"Arial","host",1);
pdf_setfont($pdf,$arial,$pointSize);
```

Con un ciclo `foreach`, ogni parola contenuta nell'array `$words` viene misurata e aggiunta alla fine di una frase. Quando la frase dovrà essere più lunga rispetto alla larghezza della pagina, viene memorizzata in un array chiamato `lines` e viene iniziata una nuova frase. Il risultato è che le singole parole dell'array `$words` vengono combinate per formare un array di righe. Ciascuna frase è più corta rispetto alla larghezza di riga massima della pagina:

```
$lines = array("");
$line=0;
foreach($words as $word){
   if(pdf_stringwidth($pdf,$lines[$line] . " " . $word) < $lineWidth){
      $lines[$line] = $lines[$line] . $word . " ";
   }
   else {
      $line++;
      $lines[$line] = $word . " ";
   }
}
```

Infine, a partire dall'inizio della pagina, tutte le righe di testo contenute nell'array `$lines` vengono visualizzate sulla pagina, che poi viene chiusa e salvata:

```
pdf_set_text_pos($pdf,$lrMargin,$pageHeight-$tbMargin);

foreach($lines as $aline) {
   pdf_show_xy($pdf,$aline,$lrMargin,$textPos);
   $textPos=$textPos-($pointSize+2);
}

pdf_end_page($pdf);
pdf_close($pdf);

?>
<a href="pdf/SinglePara.pdf">Visualizza il documento PDF generato.</a>
```

L'output di questo script è quello della Figura 32.4. Lo script sembra funzionare molto bene, benché presenti alcuni svantaggi. Per esempio, lo script è in grado di gestire un solo paragrafo di testo e non ha alcun mezzo per separare i paragrafi. Inoltre, se il paragrafo dovesse risultare molto lungo, potrebbe scomparire oltre il fondo della pagina, in quanto lo script non ha alcun modo per aggiungere pagine al documento. È quindi il momento di pensare a uno script in grado di risolvere questi problemi.

Figura 32.4
Output di un singolo paragrafo.

Impaginazione di un documento di testo

Lo script che segue è una modifica del precedente e lo migliora aggiungendo funzionalità per la gestione di paragrafi e pagine. Lo script inizia dichiarando due stringhe di testo. Osservate il carattere `"\n"`, che sarà utilizzato come mezzo per rilevare l'inizio di un nuovo paragrafo. Le due stringhe sono concatenate assieme diverse volte per dar vita a una grande stringa chiamata `$text`.

```
<?php

// Documenti PDF - Esempio 32-5
//-------------------------------

$text1="Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa. ";

$text2="\n Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa.
        Sopra la panca la capra campa, sotto la panca la capra crepa. ";

$text = $text1 . $text2 . $text2 . $text2 . $text2 . $text2 . $text2 .
        $text2;
```

La sezione seguente dello script è invariata rispetto all'esempio precedente:

```
$pointSize=32;
$pageWidth=595;
$pageHeight=842;
$lrMargin=80;
$tbMargin=80;
$lineWidth=$pageWidth-($lrMargin*2);
$textPos=$pageHeight-$tbMargin;

$words = explode(" ",$text);

$pdf = pdf_new();
pdf_open_file($pdf,"pdf/paragraphs.pdf");
pdf_begin_page($pdf,$pageWidth,$pageHeight);
$arial = pdf_findfont($pdf,"Arial","host",1);
pdf_setfont($pdf,$arial,$pointSize);
```

Quest'altra sezione dello script genera le righe di testo da visualizzare sulla pagina. È molto simile allo script precedente, ma ora contiene un'istruzione `if` aggiuntiva nel ciclo `foreach`, che controlla la presenza di un carattere `"\n"`. Se lo trova, il carattere `"\n"` viene inserito nell'array `$lines` per indicare che dev'essere visualizzata una riga vuota:

```
$lines = array("");
$line=0;
foreach($words as $word){
   if($word == "\n"){
      $line++;
      $lines[$line] = $word;
      $line++;
      $lines[$line] = "";
   }
   elseif(pdf_stringwidth($pdf,$lines[$line] . " " . $word) < $lineWidth){
      $lines[$line] = $lines[$line] . $word . " ";
   }
   else {
      $line++;
      $lines[$line] = $word . " ";
   }
}

pdf_set_text_pos($pdf,$lrMargin,$pageHeight-$tbMargin);
```

Il ciclo `foreach` che segue, che visualizza le righe di testo sulla pagina, ha subito due modifiche. Innanzitutto viene verificata la presenza di una riga costituita da `"\n"`. Se viene trovata, la variabile `$textPos` che viene utilizzata per memorizzare la posizione orizzontale del testo è ridotta di una riga di testo; questo ha l'effetto di creare uno spazio tra i paragrafi:

```
foreach($lines as $aline) {
   if($aline == "\n"){
      $textPos=$textPos-($pointSize+2);
   }
   else {
      pdf_show_xy($pdf,$aline,$lrMargin,$textPos);
      $textPos=$textPos-($pointSize+2);
   }
```

Inoltre, viene controllato se la variabile `$textPos` è scesa al di sotto della posizione minima sulla pagina sulla quale si desidera visualizzare il testo. Se ciò accade, la pagina esistente viene chiusa e se ne apre una nuova per consentire la visualizzazione del testo:

```
   if($textPos <= $tbMargin){
      $textPos=$pageHeight-$tbMargin;
      pdf_end_page($pdf);
      pdf_begin_page($pdf,$pageWidth,$pageHeight);
      $arial = pdf_findfont($pdf,"Arial","host",1);
      pdf_setfont($pdf,$arial,$pointSize);
      pdf_set_text_pos($pdf,$lrMargin,$pageHeight-$tbMargin);
   }
}
```

```
pdf_end_page($pdf);
pdf_close($pdf);

?>
<a href="pdf/paragraphs.pdf">Visualizza il documento PDF generato.</a>
```

L'output prodotto da questo script è quello delle Figure 32.5 e 32.6.

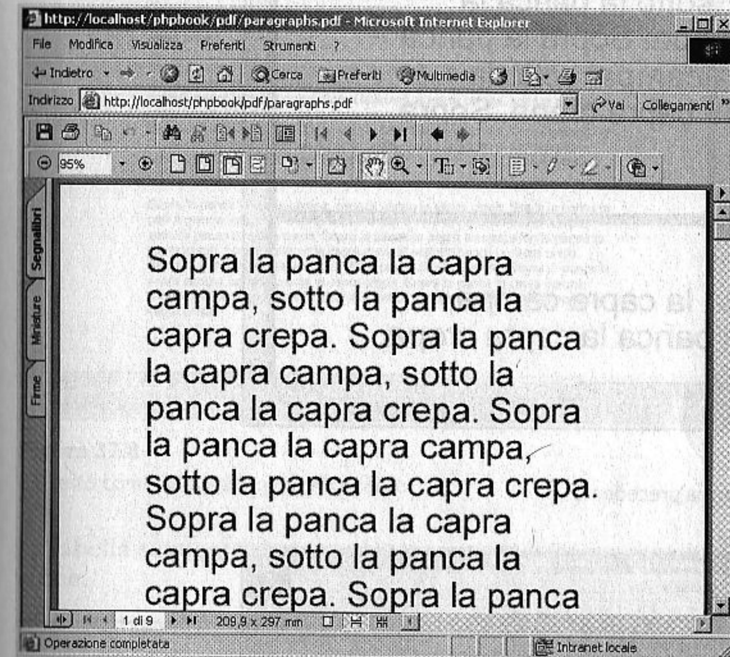


Figura 32.5
Impaginazione di paragrafi con un font da 16 punti ...

Osservate che il testo appare su più pagine. Siete liberi di regolare la dimensione per vedere se il testo appare correttamente. Le Figure 32.7 e 32.8 illustrano l'output prodotto dallo stesso script con la dimensione del font ridotta a 12 punti.

Inserimento di immagini

Per poter inserire immagini, è necessario introdurre diverse funzioni nuove, la prima delle quali è `pdf_open_image_file()`, che consente di specificare l'immagine che si intende incorporare nel PDF.
La sintassi della funzione è la seguente.

int pdf_open_image_file(int *pdfHandle*, string *tipoImmagine*, string *nomeImmagine*)

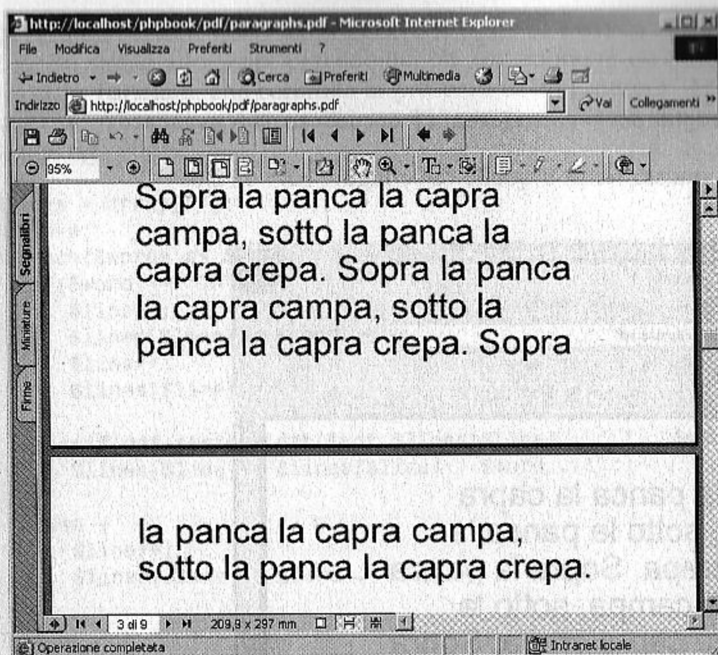


Figura 32.6
... il testo continua dalla pagina precedente.

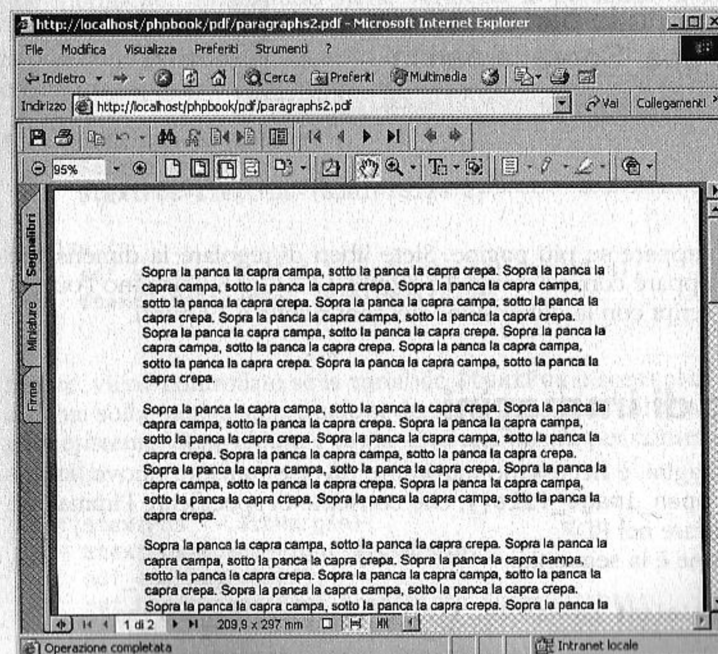


Figura 32.7
Impaginazione di paragrafi con un font da 12 punti ...

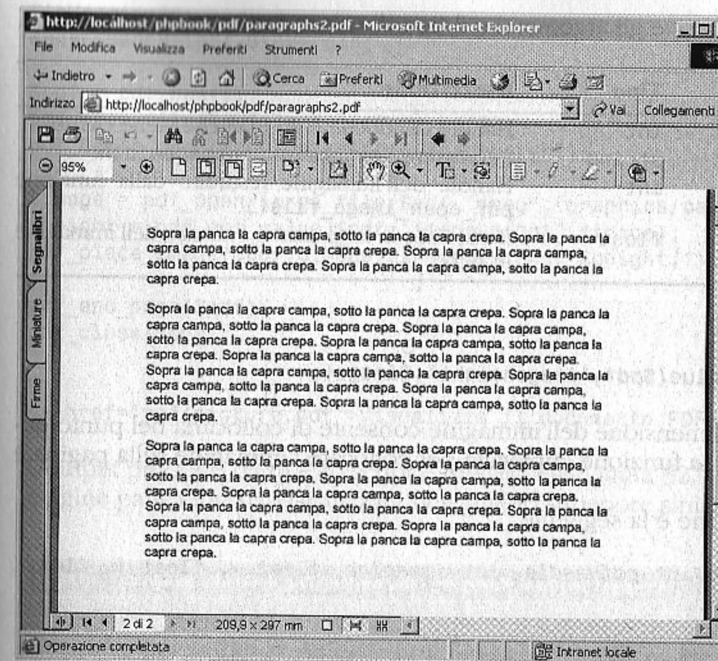


Figura 32.8
... il testo continua dalla pagina precedente.

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
tipoImmagine	string	Tipo di file di immagine, che può essere png, gif, jpeg o tiff.
nomeImmagine	string	Nome del file di immagine.
Restituzione di `pdf_open_image_file()`	int	La funzione restituisce un handle per l'immagine.

Esempio di funzione:

```
$image = pdf_open_image_file($pdf,"jpeg","palma.jpg");
```

Una volta selezionata l'immagine che si desidera utilizzare, è necessario determinarne la dimensione. `pdf_get_value()` è una funzione piuttosto versatile che restituisce i valori in una grande quantità di elementi in relazione ai parametri che le vengono forniti.
La sintassi della funzione è la seguente.

```
float pdf_get_value(int pdfHandle, string parametro, int immagine)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
parametro	string	Può essere imageheight o imagewidth.
immagine	int	Handle dell'immagine restituito dalla funzione pdf_open_image_file().
Restituzione di *pdf_get_value()*	float	Restituisce la larghezza o l'altezza dell'immagine in punti.

Esempio di funzione:

```
$height = pdf_get_value($pdf,"imageheight",$image);
```

La conoscenza della dimensione dell'immagine consente di collocarla nel punto desiderato nella pagina. La funzione impiegata per collocare l'immagine sulla pagina è `pdf_place_image()`.
La sintassi della funzione è la seguente.

```
void pdf_place_image(int pdfHandle, int immagine, float x, float y, float
scala)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
immagine	int	Handle dell'immagine restituito dalla funzione pdf_open_image_file().
x	float	La coordinata X sulla pagina in cui verrà collocato l'angolo inferiore sinistro dell'immagine.
y	float	La coordinata Y sulla pagina in cui verrà collocato l'angolo inferiore sinistro dell'immagine.
scala	float	Il fattore di scala. Un valore 1 visualizzerà l'immagine a grandezza piena, 0.5 la visualizzerà con dimensioni dimezzate.
Restituzione di pdf_place_image()	void	Non restituisce alcunché.

Esempio di funzione:

```
pdf_place_image($pdf,$image,100,100,1);
```

Lo script che segue illustra l'utilizzo di queste funzioni per includere un'immagine in un documento PDF. Dato che per poter includere un'immagine nel documento ce ne occorre una, dobbiamo impiegare l'immagine palma.jpg:

```
<?php

// Documenti PDF - Esempio 32-6
//-----------------------------

$pageWidth=595;
$pageHeight=842;
$lrMargin=80;
$tbMargin=80;
$pos=$pageHeight-$tbMargin;

$pdf = pdf_new();
pdf_open_file($pdf,"pdf/picture.pdf");
pdf_begin_page($pdf,$pageWidth,$pageHeight);

$image = pdf_open_image_file($pdf,"jpeg","graphics/palma.jpg");
$height = pdf_get_value($pdf,"imageheight",$image);
pdf_place_image($pdf,$image,$lrMargin,$pos-$height,1);

pdf_end_page($pdf);
pdf_close($pdf);

?>
<a href="pdf/picture.pdf">Visualizza il documento PDF generato.</a>
```

L'output prodotto da questo script è quello della Figura 32.9, che mostra come l'immagine palma.jpg sia visualizzata nell'angolo superiore sinistro della pagina.

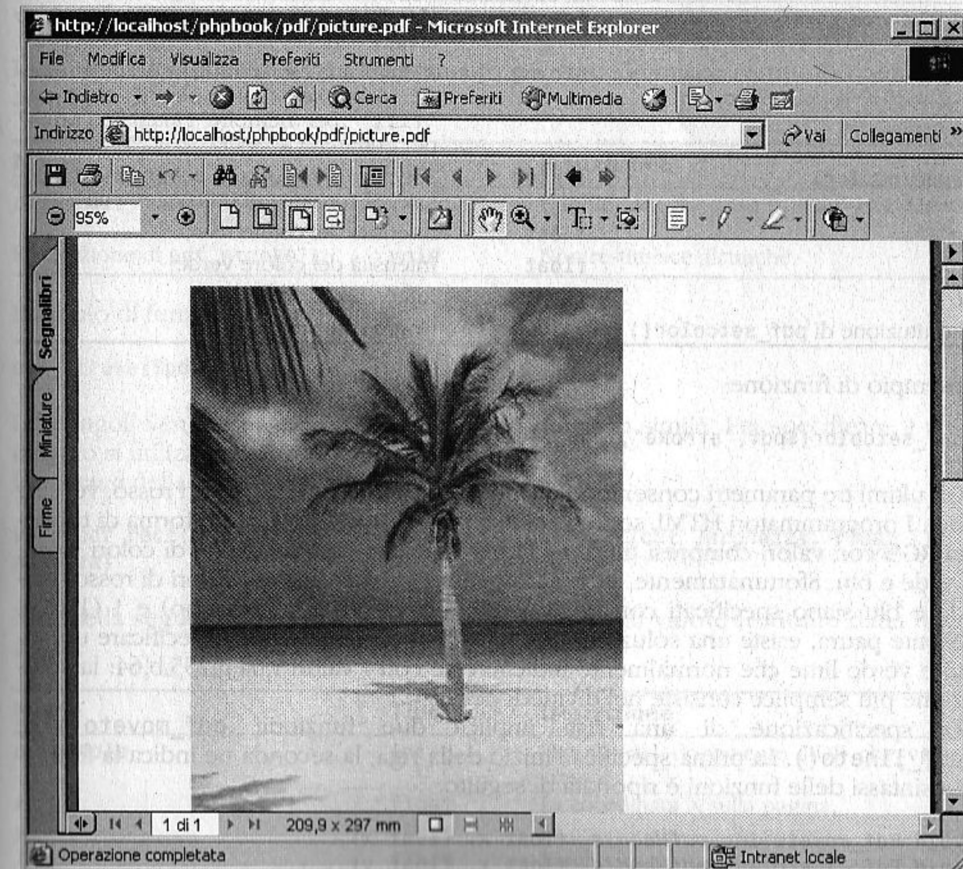


Figura 32.9
Un'immagine in un documento PDF.

Creazione di proprie immagini

La libreria PDFlib offre una grande quantità di funzioni che consentono di creare i propri elementi grafici in un documento PDF, tra cui forme e righe. Per fortuna, una volta compreso come creare un tipo di immagine, la creazione degli altri è un processo molto simile e richiede semplicemente la ricerca della funzione appropriata. Dopo aver letto queste pagine sarete in grado di creare righe e rettangoli. In PDF, la creazione di forme implica tre operazioni principali: definizione del colore, definizione della forma da visualizzare e visualizzazione di quest'ultima. La funzione `pdf_setcolor()` serve per definire un colore.
La sintassi della funzione è la seguente.

```
void pdf_setcolor(int pdfHandle, string tipo, string spazioColori, float
r, float g, float b)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
tipo	`string`	Identifica la modalità di utilizzo del colore: `"stroke"` solo per il contorno, `"fill"` per riempire l'interno, `"both"` per entrambe le operazioni .
spazioColori	`string`	Specifica il metodo di definizione del colore. Si consiglia di utilizzare `"rgb"`.
r	`float`	Intensità del colore rosso.
g	`float`	Intensità del colore verde.
b	`float`	Intensità del colore blu.
Restituzione di `pdf_setcolor()`	`void`	Non restituisce alcunché.

Esempio di funzione:

```
pdf_setcolor($pdf,"stroke","rgb",0,0,255/255);
```

Gli ultimi tre parametri consentono di specificare l'intensità dei colori rosso, verde e blu. I programmatori HTML sono abituati a specificare i colori sotto forma di tripletta RGB con valori compresi tra 0 e 255 per ciascuna combinazione di colori rosso, verde e blu. Sfortunatamente, `pdf_setcolour()` richiede che i valori di rosso, verde e blu siano specificati con un'intensità compresa tra 0 (nessuno) e 1 (100%). Niente paura, esiste una soluzione semplice. Supponete di voler specificare un colore verde lime che normalmente indichereste con i valori r,64;g,255.b,64: la soluzione più semplice consiste nel dividerli per 255.
La specificazione di una riga implica due funzioni: `pdf_moveto()` e `pdf_lineto()`. La prima specifica l'inizio della riga, la seconda ne indica la fine.
La sintassi delle funzioni è riportata di seguito.

```
void Pdf_moveto(int pdfHandle, float x, float x)
void Pdf_lineto(int pdfHandle, float x, float y)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalle funzioni.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
x	`float`	La coordinata X sulla pagina.
y	`float`	La coordinata Y sulla pagina.
Restituzione di `pdf_moveto()` e `pdf_lineto()`	`void`	Non restituiscono alcun particolare valore.

Esempio di funzione:

```
pdf_moveto($pdf,$lrMargin,100);
pdf_lineto($pdf,$pageWidth-$lrMargin,100);
```

Infine, per disegnare la riga si utilizza la funzione `pdf_stroke()`.
La sintassi della funzione è la seguente.

```
void pdf_stroke(int pdfHandle)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
Restituzione di `pdf_stroke()`	`void`	Non restituisce alcunché.

Esempio di funzione:

```
pdf_stroke($pdf);
```

I rettangoli vengono specificati e visualizzati in modo simile. Per specificare il rettangolo si utilizza la funzione `pdf_rect()`.
La sintassi della funzione è la seguente.

```
void pdf_rect(int pdfHandle, float x, float y, float larghezza, float
altezza)
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	`int`	Handle del documento PDF che viene creato.
x	`float`	La coordinata X sulla pagina.
y	`float`	La coordinata Y sulla pagina.
larghezza	`float`	Larghezza del rettangolo.
altezza	`float`	Altezza del rettangolo.
Restituzione di `pdf_rect()`	`void`	Non restituisce alcunché.

Esempio di funzione:

```
pdf_rect($pdf,$lrMargin+10,110,$pageWidth-($lrMargin*2)-20,$pageHeight-
220);
```

Per visualizzare il rettangolo sulla pagina si utilizza la funzione `pdf_fill_stroke()`.
La sintassi della funzione è la seguente.

```
void pdf_fill_stroke(int pdfHandle)
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
pdfHandle	int	Handle del documento PDF che viene creato.
Restituzione di `pdf_fill_stroke()`	void	Non restituisce alcunché.

Esempio di funzione:

```
pdf_fill_stroke($pdf);
```

Lo script seguente illustra l'utilizzo di queste funzioni con un semplice esempio. Lo script produce un documento costituito da una pagina con due righe in alto e in basso e un grande rettangolo rosa:

```
<?php

// Documenti PDF - Esempio 32-7
//-------------------------------

$pageWidth=595;
$pageHeight=842;
$lrMargin=80;
$tbMargin=80;
$pos=$pageHeight-$tbMargin;

$pdf = pdf_new();
pdf_open_file($pdf,"pdf/linerect.pdf");
pdf_begin_page($pdf,$pageWidth,$pageHeight);

pdf_setcolor($pdf,"stroke","rgb",0,0,255/255);
pdf_moveto($pdf,$lrMargin,100);
pdf_lineto($pdf,$pageWidth-$lrMargin,100);
pdf_stroke($pdf);

pdf_moveto($pdf,$lrMargin,$pageHeight-100);
pdf_lineto($pdf,$pageWidth-$lrMargin,$pageHeight-100);
pdf_stroke($pdf);

pdf_setcolor($pdf,"both","rgb",233/255,157/255,157/255);
pdf_rect($pdf,$lrMargin+10,110,$pageWidth-($lrMargin*2)-20,$pageHeight-220);
pdf_fill_stroke($pdf);

pdf_end_page($pdf);
pdf_close($pdf);

?>
<a href="pdf/linerect.pdf">Visualizza il documento PDF generato.</a>
```

L'output di questo script è quello della Figura 32.10.

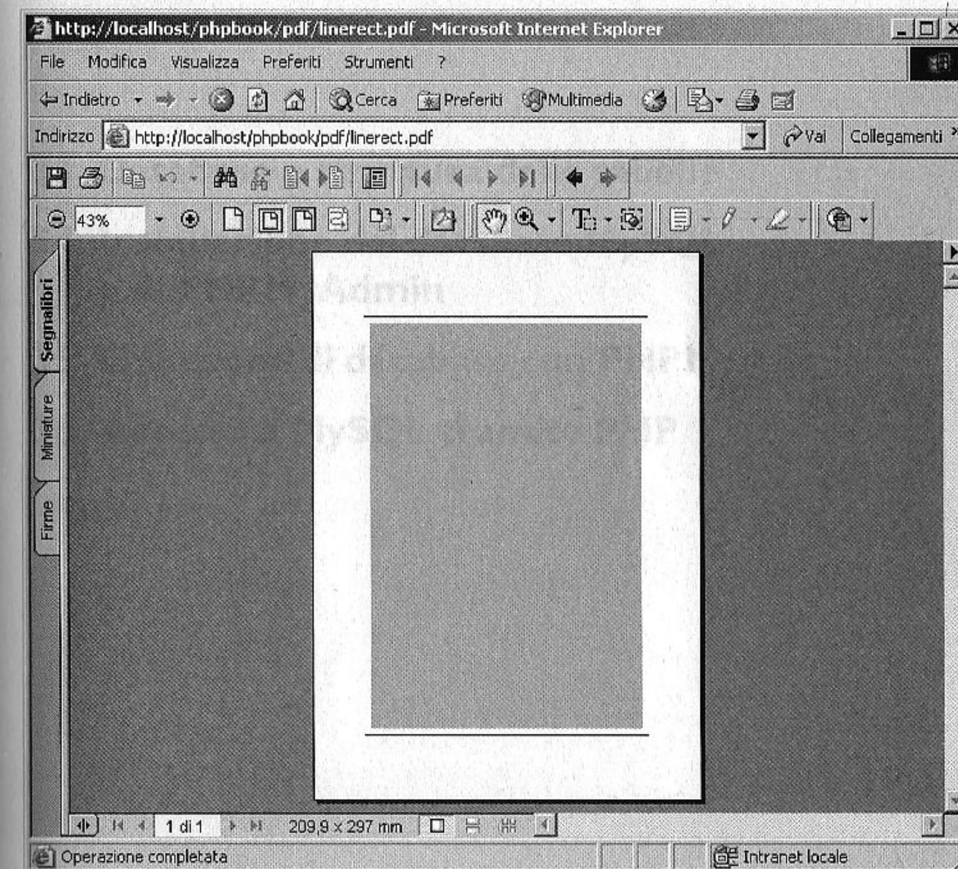


Figura 32.10
Rettangolo e righe.

Riepilogo

Questo capitolo ha introdotto alcune funzioni che fanno parte della libreria PDFLib. Si è visto come sia possibile creare semplici documenti PDF e come superare alcune difficoltà poste dalla formattazione di questi documenti. Il prossimo capitolo si occuperà di database relazionali e della progettazione di tabelle.

Capitolo 33

Database e struttura delle tabelle

Introduzione

In questo capitolo verranno illustrati i sistemi di gestione di database relazionali, i database, le tabelle, i campi delle colonne e i record. Nei punti più appropriati si farà riferimento al sistema di gestione di database MySQL, poiché è quello che sarà utilizzato più avanti per gestire i database.

Definizione di database

Un database è una collezione strutturata di dati, che in quanto tale esisteva molto prima dell'invenzione del computer. Ecco alcuni esempi di database del "mondo reale":

- una guida con i programmi televisivi;
- un classificatore per l'archiviazione di documenti;
- una rubrica telefonica.

Un database informatico è uno strumento per memorizzare in forma strutturata informazioni che potranno essere recuperate e analizzate velocemente e facilmente. Ecco alcuni esempi di database informatici:

- DVLA (*Driver and Vehicle Licensing Agency*) che registra le informazioni relative a tutti i veicoli immatricolati nel Regno Unito;
- HOLMES (*Home Office Large Major Enquiry System*) che memorizza i dati relativi a criminali e persone sospette coinvolti in investigazioni attuali e passate della polizia;
- Amazon.com, un database di libri da vendere collegato a un sito di commercio elettronico.

Tutti questi database sono molto grandi, ma il concetto rimane sempre lo stesso a prescindere dalle dimensioni.

Sistema di gestione di database

Un sistema di gestione di database, o DBMS (*Database Management System*), è il software che agevola la creazione e la gestione di un database informatico. In generale, un DBMS esegue i seguenti compiti:

- gestisce una grande quantità di dati;
- fornisce l'accesso ai dati con un linguaggio di interrogazione;
- fornisce una certa protezione per i dati;
- consente accessi multipli al database.

MySQL è un DBMS relazionale. Un database relazionale memorizza i dati in tabelle separate invece che in un unico archivio. In teoria, questa soluzione dà vita a un sistema di database più veloce e flessibile.

Tabella di database?

Una tabella di database è molto simile a una tabella inserita in un documento di elaborazione testi, poiché è composta da righe e colonne. Spesso le colonne prendono anche il nome di "campi" e sono utilizzate per delimitare la struttura dei dati nell'ordine corretto. Nelle righe di una tabella del database sono memorizzati i record. All'interno di un database specifico, a una tabella viene assegnato un nome univoco.
La Figura 33.1 illustra una semplice tabella di database.

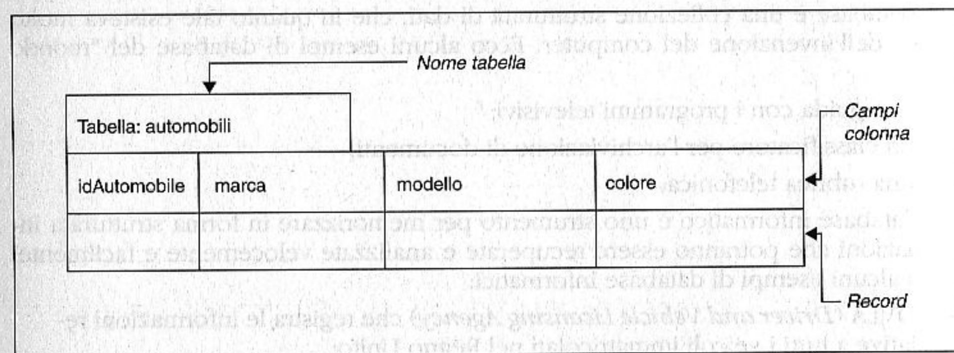


Figura 33.1
Tabella di database.

Nella Figura 33.1 potete vedere che la tabella del database si chiama "automobili" ed è costituita da quattro campi: idAutomobile, marca, modello e colore. Nella tabella non ci sono record di dati.

Campi del database

I campi del database definiscono la struttura dei dati contenuti in una tabella. Come le variabili, anche i campi del database possono essere di tipi diversi. A complicare leggermente la situazione contribuisce il fatto che i tipi di campo non sempre sono gli stessi del linguaggio di programmazione utilizzato per accedere al database. Nel sistema di gestione di database MySQL è possibile definire tipi di campo diversi. Questi tipi di campo sono elencati nelle Tabelle 33.1-33.3.

Tabella 33.1 Tipi di campo testuali

Tipo	Lunghezza massima	Descrizione
`varchar`	255 caratteri	Campo di testo a lunghezza variabile.
`char`	255 caratteri	Campo di testo a lunghezza fissa.
`tinytext`	255 caratteri	Campo di testo a lunghezza variabile.
`text`	65.535 caratteri	Campo di testo a lunghezza variabile.
`mediumtext`	16.777.215 caratteri	Campo di testo a lunghezza variabile.
`longtext`	4.294.967.295 caratteri	Campo di testo a lunghezza variabile.
`enum`	65.535 caratteri	Valori potenziali di un campo di testo.

Tabella 33.2 Tipi di campo numerici

Tipo	Dimensione numerica	Descrizione
`int`	4.294.967.295	Campo numerico con o senza segno.
`tinyint`	255	Campo numerico con o senza segno.
`mediumint`	16.777.215	Campo numerico con o senza segno.
`bigint`	18.446.744.073.709	Campo numerico con o senza segno.
`float`	-	Campo numerico in virgola mobile con segno.
`double`	-	Campo numerico in virgola mobile con segno.
`decimal`	-	Campo numerico con segno (numeri memorizzati come caratteri).

Tabella 33.3 Tipi di campo data e ora

Tipo	Valori	Descrizione
`date`	Dal 01-01-1001 al 31-12-9999	Memorizza i dati in un formato YYYY-MM-DD (anno-mese-giorno)
`time`	Da –838:59:59 a 838:59:59	Memorizza i dati in un formato ora
`datetime`	Dalle 00:00:00 del 01-01-1001 alle 23:59:59 del 31-12-9999	Memorizza i dati in un formato YYYY-MM-DD HH:MM:SS (anno-mese-giorno ore:minuti:secondi)
`timestamp`	2-14	Memorizza i valori numerici per presentare diversi tipi di indicatori data e ora UNIX
`year`	dal 1901 al 2155	Memorizza quattro cifre (o due) per presentare l'anno.

La Figura 33.2 mostra la tabella del database modificata in modo da specificare il tipo di ciascun campo.

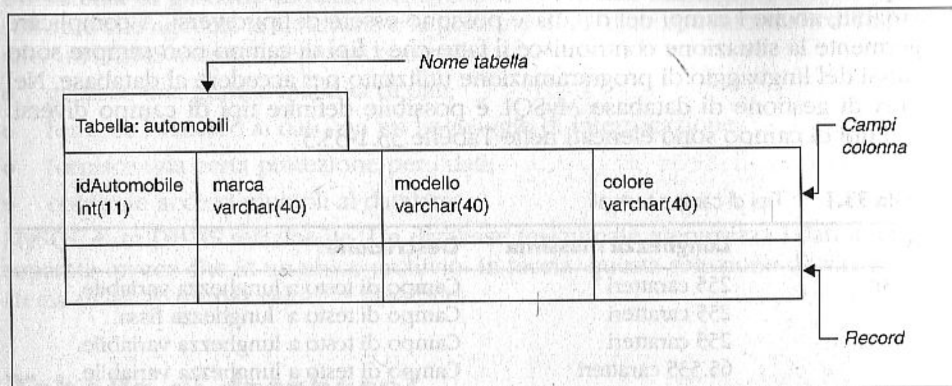


Figura 33.2
Tabella del database con i tipi di campi.

Il campo *idAutomobile* è di tipo intero e può memorizzare numeri di 11 cifre. I campi *marca*, *modello* e *colore* sono tutti del tipo `varchar`, con una dimensione massima di 40 caratteri.

Attributi speciali dei campi e chiavi

Oltre al tipo, per un campo è possibile specificare anche alcuni attributi speciali, elencati nella Tabella 33.4. Tali attributi descrivono alcune proprietà speciali del campo.

Tabella 33.4 Attributi speciali

Attributo	Descrizione
`autoincrement`	Il valore del campo viene creato automaticamente insieme a un nuovo record. Tale record non è fornito dall'utente. Questo attributo serve soprattutto a garantire la creazione di un valore di campo univoco, poiché il valore prodotto è maggiore di uno rispetto all'ultimo creato.
`not null`	Il campo del database non può essere vuoto. Se è vuoto, viene generato un errore.
`default`	Questo attributo specifica il valore cui verrà impostato il campo se non si fornisce alcun valore.

Campi univoci

È possibile specificare un campo come univoco. In altre parole, in una tabella non ci possono essere due record con lo stesso valore nel campo. Un esempio potrebbe essere una tabella contenente un elenco con nomi di persone e codici fiscali. Anche se due persone possono avere lo stesso nome, non possono avere il medesimo codice fiscale.

Chiavi

I campi di una tabella possono essere specificati come chiavi primarie. Una chiave primaria serve a creare un indice dei record del database all'interno di una tabella. La chiave primaria dev'essere anch'essa univoca, tuttavia oltre a ciò viene generato un indice della tabella che consente ricerche e recuperi di record molto più rapidi.

Record del database

I record del database (noti anche con il nome di righe) formano le righe della tabella del database. I record devono corrispondere al tipo e agli attributi speciali di ciascun campo. La Figura 33.3 mostra la tabella del database con tre record.

Nome tabella

Tabella: automobili

Campi colonna

idAutomobile Int(11)	marca varchar(40)	modello varchar(40)	colore varchar(40)
1	Ford	Focus	Blu
2	Ford	Fiesta	Verde
3	Mazda	323	Nera

Record

Figura 33.3
Record della tabella del database.

Riepilogo

Questo capitolo ha spiegato che cos'è un database e un sistema di gestione di database. Ha illustrato inoltre il concetto di tabella del database e descritto i vari tipi di campi disponibili in MySQL. Nel prossimo capitolo vedrete come installare il sistema di gestione del database MySQL e il software PHPMyAdmin.

Capitolo 34

Installazione del database MySQL e di PHPMyAdmin

Introduzione

Questo capitolo descrive come procurarsi, installare e configurare il sistema di gestione di database MySQL. Stando a quanto afferma il sito Web di MySQL, "MySQL è il server di database open source più diffuso, con oltre quattro milioni di installazioni a supporto di siti Web, banche dati, applicazioni aziendali, sistemi di registrazione e altro ancora. Organizzazioni come Yahoo!, Finance, MP3.com, Motorola, NASA, Silicon Graphics e Texas Instruments utilizzano il server MySQL per applicazioni mission-critical". Uno tra i maggior pregi di MySQL è la possibilità di scaricarlo e utilizzarlo gratuitamente.

Benché MySQL sia un database molto potente con numerose funzionalità (normalmente offerte solo da prodotti commerciali molto costosi), è penalizzato dal fatto di non offrire un'interfaccia grafica di agevole utilizzo. Tuttavia non tutto è perduto, in quanto il progetto PHPMyAdmin offre uno strumento molto sofisticato e di facile impiego, scritto in PHP, concepito per gestire l'amministrazione di MySQL attraverso il World Wide Web. In queste pagine si farà ricorso sia a MySQL sia a PHPMyAdmin per creare le applicazioni di database utilizzate negli esempi. Per cominciare imparerete come procurarvi MySQL.

Scaricamento di MySQL

MySQL può essere scaricato gratuitamente dal sito `www.mysql.com`. È preferibile scaricare l'ultima release stabile da `http://www.mysql.com/downloads/mysql-4.0.html`. Esistono molte versioni per diversi sistemi operativi, quindi dovete accertarvi di selezionare quella giusta.

Se desiderate scaricare MySQL per utilizzarlo su una piattaforma Windows, potete scegliere di procurarvi un pacchetto di applicazioni PHP completo presso l'AppServ Open Project all'indirizzo `http://appserv.sourceforge.net/`. Questo pacchetto include l'ultima versione stabile di MySQL e PHPMyAdmin. Nel Capitolo 3 sono stati discussi ulteriori dettagli relativi a questo pacchetto di applicazioni.

Installazione del server MySQL (su server Linux/UNIX)

Se state installando MySQL su una piattaforma Windows, o se il provider di servizi o l'amministratore di sistema hanno già installato e configurato MySQL, potete ignorare questo paragrafo.

Istruzioni da seguire per installare il database MySQL su un server UNIX.

1. Accedete al server con il nome utente "root".
2. Scompattate tutti i file sorgente:
   ```
   [root@server root]# tar -xzf mysql-4.0.13.tar.gz
   ```
3. Cambiate la directory di lavoro corrente in *mysql-4.0.13*:
   ```
   [root@server root]# cd mysql-4.0.13
   ```
4. Configurate il server MySQL:
   ```
   [root@server root]# ./configure --prefix=/usr/local/mysql
   ```
5. Compilate e installate il server MySQL:
   ```
   [root@server root]# make
   [root@server root]# make install
   ```
6. Installate il database MySQL principale:
   ```
   [root@server root]# ./scripts/mysql_install_db
   ```
7. Create un link simbolico all'eseguibile `mysqld`:
   ```
   [root@server root]# ln -s /usr/local/mysql/share/mysql/
   mysql.server /sbin/mysqld
   ```
8. Create un gruppo `mysql`:
   ```
   [root@server root]# groupadd mysql
   ```
9. Create un utente `mysql`:
   ```
   [root@server root]# useradd -g mysql mysql
   ```
10. Create i proprietari appropriati:
    ```
    [root@server root]# chown -R mysql:mysql /usr/local/mysql
    ```

A questo punto potete avviare, arrestare e riavviare MySQL digitando quanto segue:

```
[root@server root]# mysqld start
[root@server root]# mysqld stop
[root@server root]# mysqld restart
```

Potete disinstallare MySQL applicando le seguenti istruzioni:

```
[root@server root]# rm -rf /usr/local/mysql
[root@server root]# rm -rf /data/mysql
[root@server root]# rm -rf /sbin/mysqld
```


Installazione del server MySQL (su server Windows)

Se avete scaricato MySQL come parte dell'AppServ Open Project, o se il provider di servizi o l'amministratore di sistema hanno già installato e configurato MySQL, potete ignorare questo paragrafo.

Istruzioni da seguire per installare il database MySQL su un sistema Windows.

1. Decomprimete il file zip di MySQL in una directory temporanea.
2. Eseguite il file setup.exe e seguite le istruzioni che appaiono sul monitor.
3. L'applicazione MySQL può essere configurata per essere avviata ed eseguita automaticamente in background all'avvio del computer. Si consiglia di selezionare questa opzione.

Installazione di PHPMyAdmin

PHPMyAdmin è un'applicazione scritta con PHP ed è pertanto indipendente dalla piattaforma. Non dovete quindi preoccuparvi di procurare diverse versioni per i vari sistemi operativi. Per installare PHPMyAdmin dovete scaricare il pacchetto completo dal sito `http://www.phpmyadmin.net/`. Dovete poi decomprimere i file in una directory appropriata e modificare il file config.inc.php fornito con PHPMyAdmin in modo che le impostazioni corrispondano con quelle del database.
Per scaricare l'ultima versione di PHPMyAdmin, visitate l'URL `http://phpmyadmin.sourceforge.net/`. Benché le funzionalità siano le stesse sia per Linux, sia per Windows, dovete selezionare il file di installazione compattato in maniera compatibile con il vostro sistema operativo: per un server Linux dovrete scegliere un file in formato .tar.gz, mentre per i sistemi Windows vi occorrerà il file .zip.

Configurazione di PHPMyAdmin

1. Decomprimete tutti i file in un cartella Web (una cartella che può essere visualizzata tramite il Web), come per esempio: *C:\Inetpub\wwwroot\phpMyAdmin*.
2. Nel file config.inc.php, situato nella cartella *C:\Inetpub\wwwroot\phpMyAdmin*, cambiate le righe seguenti:

   ```
   $cfg['PmaAbsoluteUri']='http://localhost/phpMyAdmin/';

   $cfg['Servers'][$i]['auth_type']='http';

   $cfg['Servers'][$i]['user']='nomeutente';
   ```

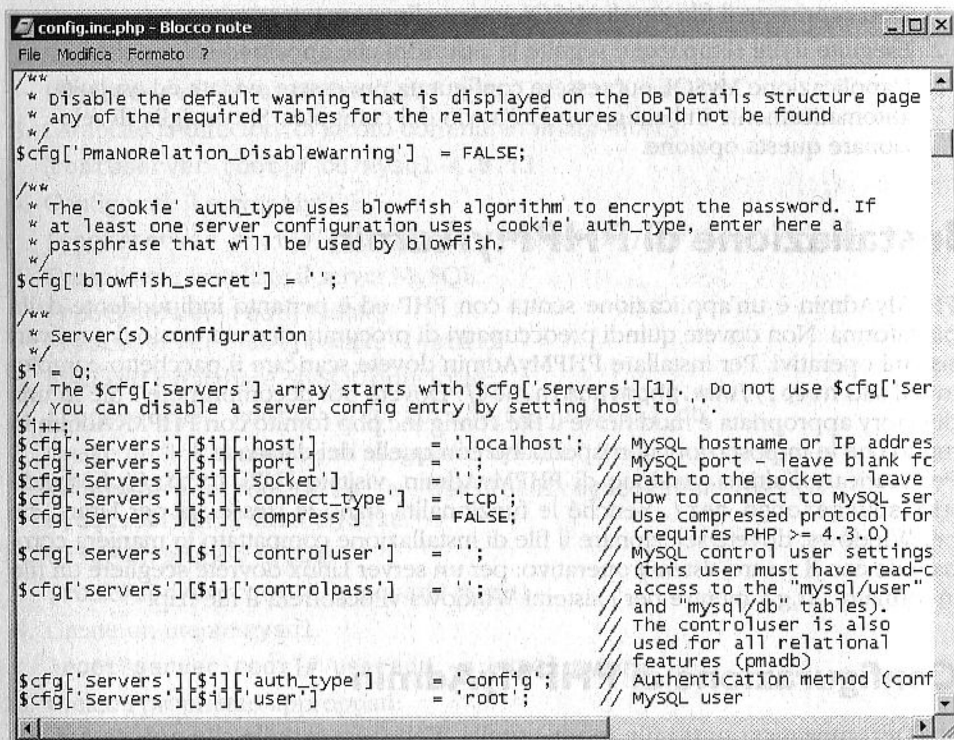
   Questa configurazione dovrebbe consentirvi di accedere e fornire un nome utente e una password. Tuttavia, nel caso in cui non funzioni, è possibile forzare un accesso con le impostazioni seguenti:

   ```
   $cfg['Servers'][$i]['auth_type']='config';
   ```

```
$cfg['Servers'][$i]['user']='root';

$cfg['Servers'][$i]['password']=";
```

Il campo password includerà la password che selezionerete più avanti nella configurazione dell'account dell'utente root. Per ora lasciatelo vuoto; sarete avvertiti quando arriverà il momento di cambiare l'impostazione. Sappiate comunque che l'inserimento di password in questo file ini è un rischio per la sicurezza del sistema. La Figura 34.1 illustra il file config.inc.php.



```
config.inc.php - Blocco note
File  Modifica  Formato  ?
/**
 * Disable the default warning that is displayed on the DB Details Structure page
 * any of the required Tables for the relationfeatures could not be found
 */
$cfg['PmaNoRelation_DisableWarning']  = FALSE;

/**
 * The 'cookie' auth_type uses blowfish algorithm to encrypt the password. If
 * at least one server configuration uses 'cookie' auth_type, enter here a
 * passphrase that will be used by blowfish.
 */
$cfg['blowfish_secret'] = '';

/**
 * Server(s) configuration
 */
$i = 0;
// The $cfg['Servers'] array starts with $cfg['Servers'][1].  Do not use $cfg['Ser
// You can disable a server config entry by setting host to ''.
$i++;
$cfg['Servers'][$i]['host']          = 'localhost'; // MySQL hostname or IP addres
$cfg['Servers'][$i]['port']          = '';          // MySQL port - leave blank fc
$cfg['Servers'][$i]['socket']        = '';          // Path to the socket - leave
$cfg['Servers'][$i]['connect_type']  = 'tcp';       // How to connect to MySQL ser
$cfg['Servers'][$i]['compress']      = FALSE;       // Use compressed protocol for
                                                    // (requires PHP >= 4.3.0)
$cfg['Servers'][$i]['controluser']   = '';          // MySQL control user settings
                                                    // (this user must have read-c
$cfg['Servers'][$i]['controlpass']   = '';          // access to the "mysql/user"
                                                    // and "mysql/db" tables).
                                                    // The controluser is also
                                                    // used for all relational
                                                    // features (pmadb)
$cfg['Servers'][$i]['auth_type']     = 'config';    // Authentication method (conf
$cfg['Servers'][$i]['user']          = 'root';      // MySQL user
```

Figura 34.1
Modifica del file config.inc.php.

Accesso a PHPMyAdmin

L'accesso a PHPMyAdmin avviene dall'URL inserito in precedenza nel file config.inc.php: `http://localhost/phpMyAdmin/index.php`. Dovreste trovarvi di fronte a una pagina Web simile a quella della Figura 34.2.

Configurazione di utente root e host in MySQL

Quando si avvia il server MySQL, vengono creati quattro utenti predefiniti. L'elenco degli utenti è visibile selezionando il collegamento *Utenti* o *Privilegi* (a seconda che

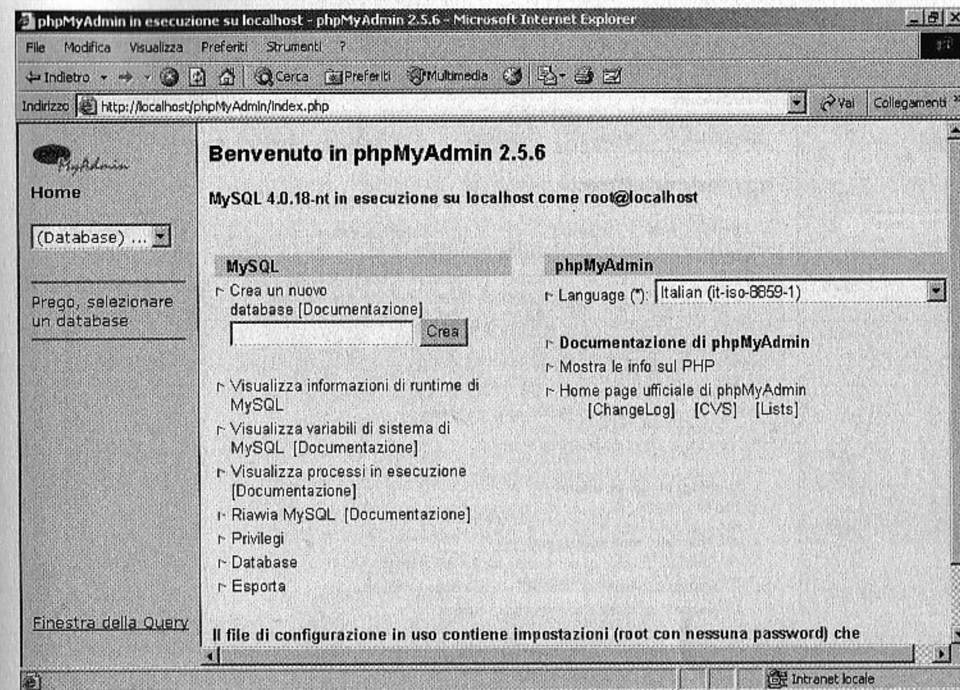


Figura 34.2
La pagina principale di PHPMyAdmin.

utilizziate Linux o Windows) dalla schermata della Figura 34.2. La schermata degli utenti è illustrata nella Figura 34.3. Dovete eliminare i primi tre utenti (come mostrato nella Figura 34.3) e lasciare solo l'utente "root" con l'host impostato su "localhost". Per togliere gli utenti, selezionate la casella di spunta accanto ai primi tre elementi dell'elenco e fate clic sul pulsante *Esegui* sotto la tabella.

Dopo aver fatto clic su questo pulsante, vi troverete di fronte alla schermata della Figura 34.4. Ora siete pronti per aggiungere una password all'utente "root". Fate scorrere verso il basso la pagina fino al punto in cui vedete *Vista d'insieme dell'utente* e fate clic sul collegamento *Modifica* accanto all'unico utente presente nella tabella, che è l'utente root. Dovreste vedere la schermata di modifica dei privilegi della Figura 34.5. Facendo scorrere verso il basso la schermata vedrete i campi del modulo nei quali dovete inserire la password (due volte). Scorrete e fate clic sul pulsante *Esegui*.

Facendo clic sulla scheda *Privilegi* in cima alla schermata tornerete all'elenco degli utenti (ora solo uno) e dovreste vedere che per l'utente è stata impostata una password (come mostrato nella Figura 34.6). A questo punto dovrete tornare indietro e modificare il file config.inc.php per includere la password che avete impostato (ma solo se avete dovuto impostare la variabile "auth_type" su "config"):

```
$cfg['Servers'][$i]['password']='password';
```

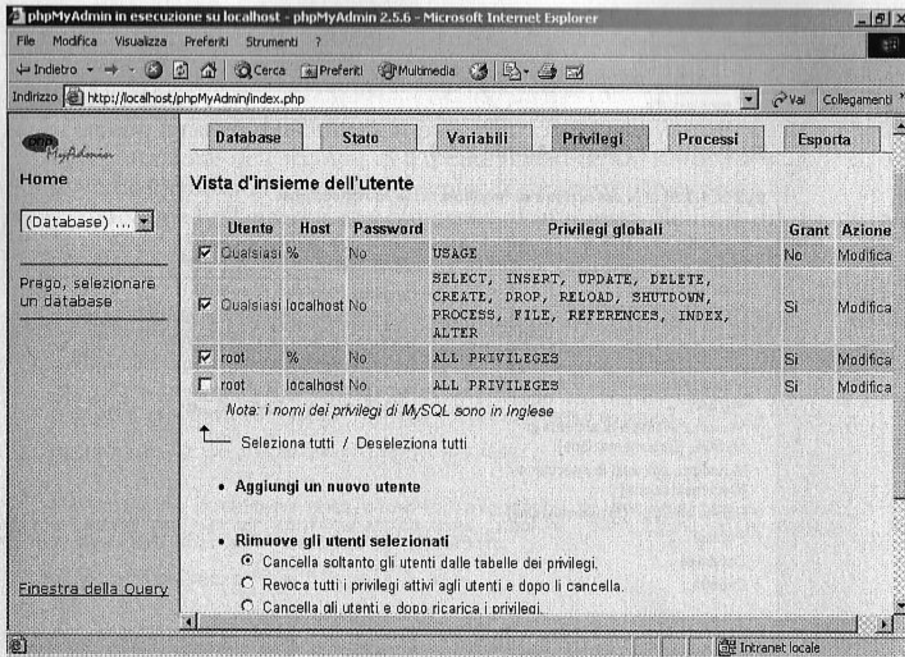


Figura 34.3
La pagina degli utenti con quattro utenti MySQL predefiniti.

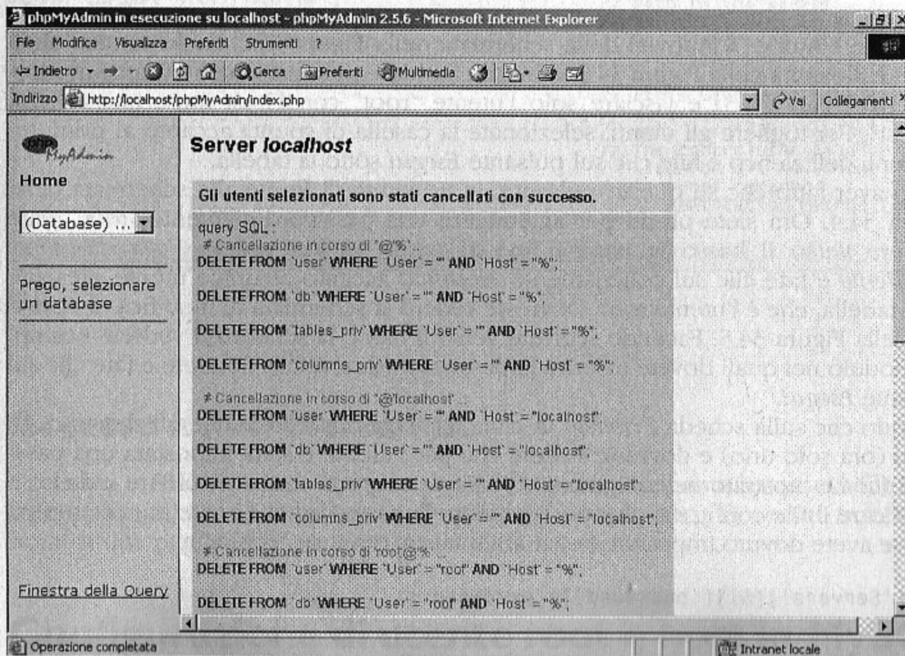


Figura 34.4
Schermata degli utenti eliminati.

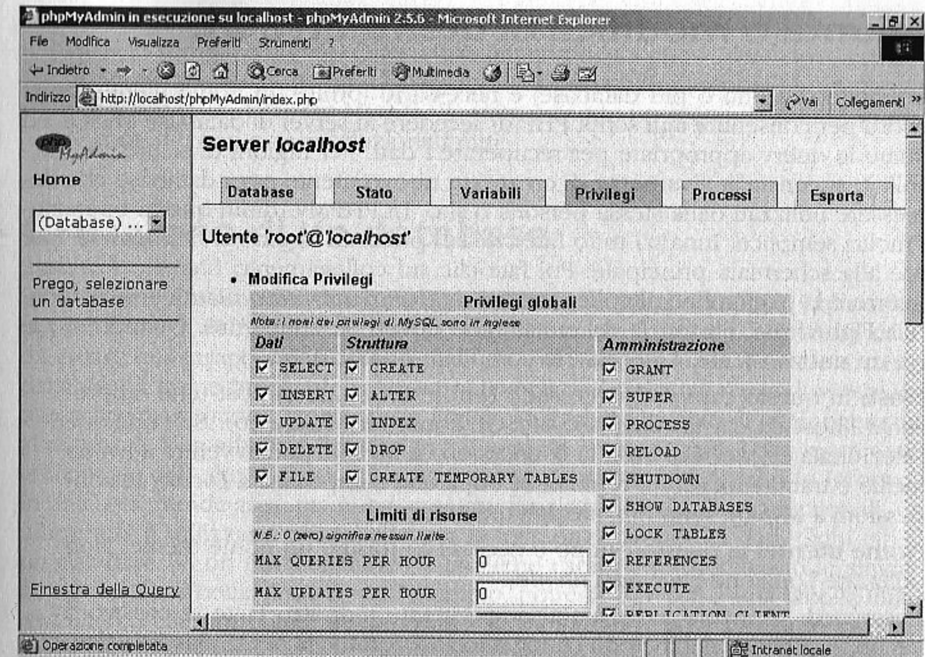


Figura 34.5
Modifica della password.

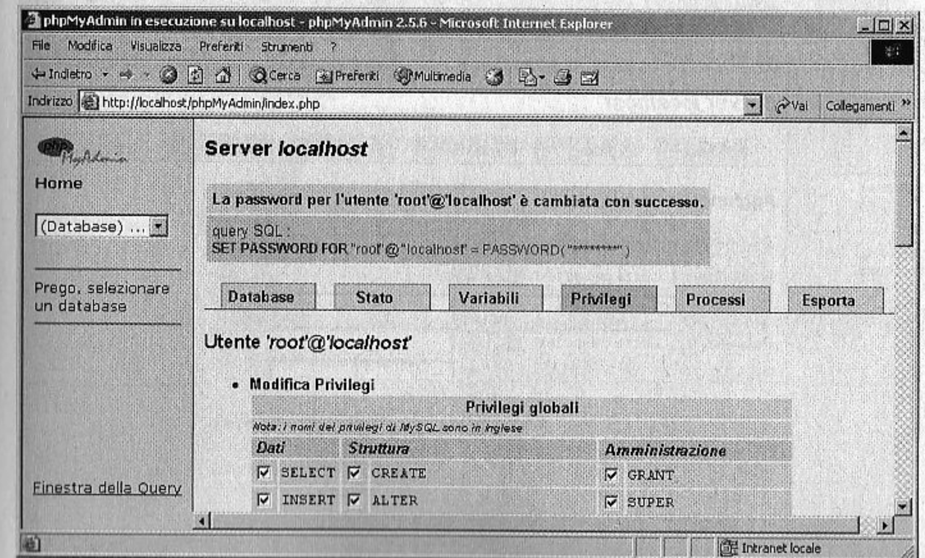


Figura 34.6
La pagina degli utenti con l'amministratore root appropriato.

Aggiunta o modifica di utenti MySQL

Per connettersi a uno o più database, è necessario fornire un nome utente e una password per consentire agli script PHP di accedere ai server di database MySQL ed effettuare le query appropriate per recuperare i dati. Per ragioni di sicurezza e accessibilità, si consiglia vivamente di creare un nuovo utente per i database che devono essere utilizzati dalla stessa persona o sito. In PHPMyAdmin questa operazione è molto semplice. Innanzi tutto fate clic sul pulsante *Aggiorna* del browser per tornare alla schermata principale. Poi fate clic sul collegamento *Utenti* o *Privilegi*, fate scorrere la pagina fino al collegamento *Aggiungi un nuovo utente* e selezionatelo. La Figura 34.7 illustra la schermata che dovrebbe comparire. Quando si aggiunge un nuovo utente, è necessario compilare quattro aree.

1. **Host:** in questo campo è necessario fornire un nome di host (che rappresenta l'host da cui un database MySQL consentirà l'accesso). Se l'opzione selezionata è *Qualsiasi host*, la connessione a MySQL può avvenire a livello locale o tramite un server remoto. Se l'opzione è impostata a *Locale*, le connessioni a MySQL sono limitate agli utenti locali.
2. **Nome utente:** in questo campo è necessario fornire un nome utente. Non utilizzate l'opzione *Qualsiasi utente*.
3. **Password:** password associata al nome utente indicato. Tutte le password vengono cifrate con il metodo crittografico MD5, tuttavia per la massima sicurezza devono essere lunghe otto o più caratteri e contenere simboli e numeri.

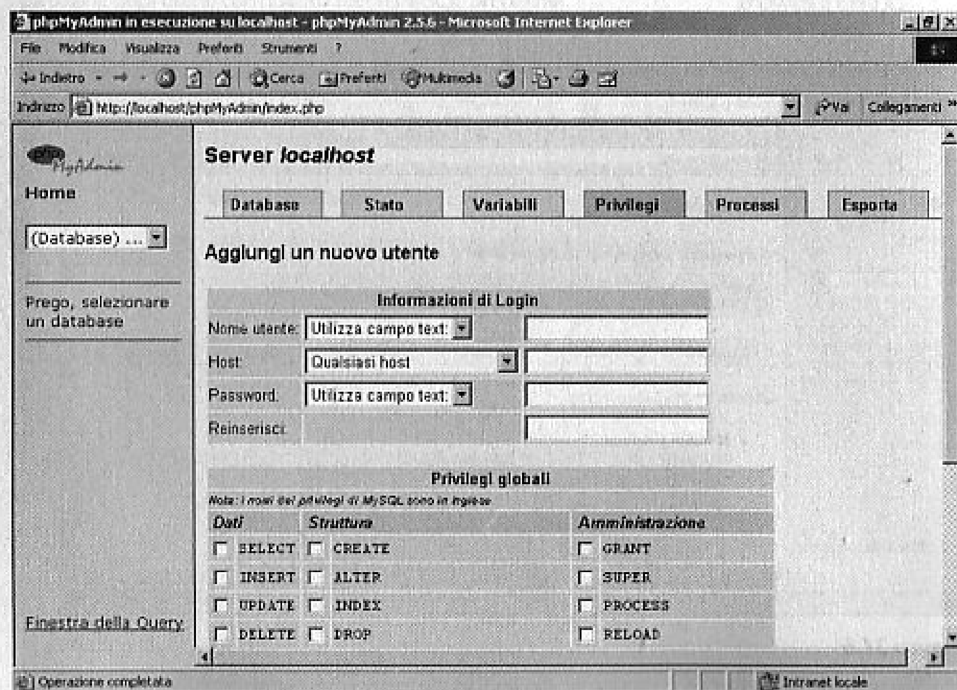


Figura 34.7
Aggiunta di un nuovo utente MySQL.

4. **Privilegi:** i privilegi devono essere sempre lasciati deselezionati quando si crea un semplice utente. Questi non hanno nulla a che vedere con i privilegi del database, in quanto riguardano le autorizzazioni globali di un utente per il server di database MySQL. Se state creando un secondo amministratore root, devono essere selezionati tutti.

Creazione di un database

La creazione di un nuovo database è un'operazione molto semplice. Dalla schermata principale di PHPMyAdmin dovete semplicemente inserire il nome del database da creare nel campo del modulo e fare clic sul pulsante *Crea*. Provate a creare un database di nome "mio_database" e poi fate clic sul pulsante *Crea*; dovreste vedere una schermata simile a quella della Figura 34.8, che consente di aggiungere tabelle al database. Una tabella è il luogo in cui vengono memorizzati i dati in un database relazionale. Gli altri sistemi di gestione dei database memorizzano le informazioni in una sola grande area, ma non è così che funzionano i database relazionali. Questi ultimi infatti utilizzano più tabelle per la memorizzazione dei dati. Ogni tabella di un database ha un nome univoco e contiene campi che descrivono le single voci dei dati della tabella. Quindi, per esempio, potreste dar vita a una tabella chiamata *Automobili* che potrebbe memorizzare i seguenti dati relativi a un'auto: marca, modello e colore. Poiché volete creare tre campi di dati, immettete `automobili` nel campo *Nome* e **3** nel campo *Campi*, quindi fate clic sul pulsante *Esegui*. PHPMyAdmin visualizzerà l'editor di tabelle della Figura 34.9.

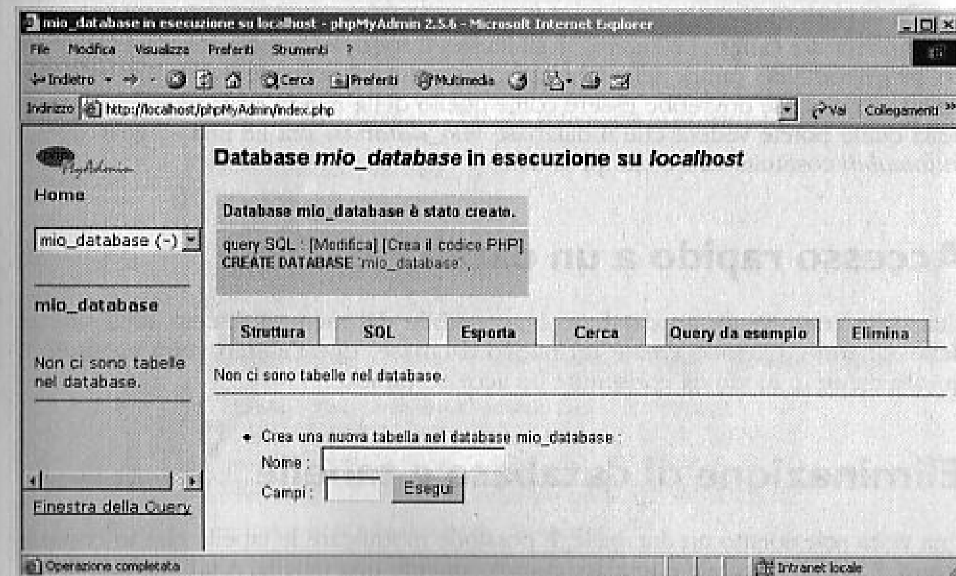


Figura 34.8
Creazione di un database.

In questa schermata è necessario attribuire un nome ai campi e determinare il tipo di dati. In relazione al tipo selezionato, potrebbe essere necessaria una lunghezza

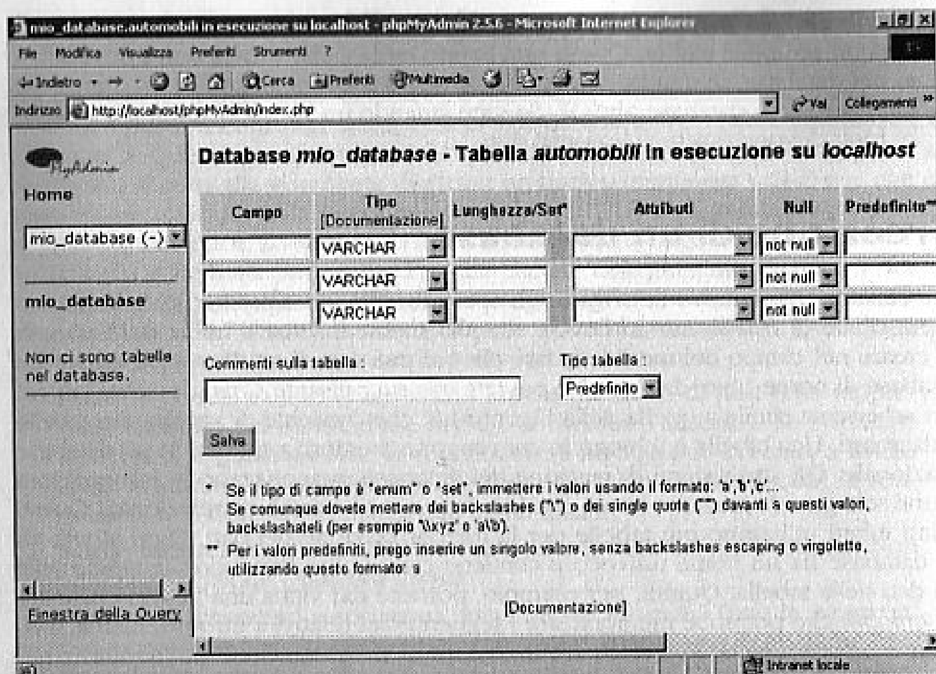


Figura 34.9
Modifica dei campi della tabella.

appropriata dei campi. La Figura 34.10 illustra il modulo completato appena prima della creazione dei campi della tabella facendo clic sul pulsante *Salva*. Una volta ultimato, il database dovrebbe essere come quello della Figura 34.11, nella quale potete vedere che il database *mio_database* ora ha una singola tabella *automobili* costituita da tre campi di dati.

Accesso rapido a un database

Una volta completato, un database è accessibile dal menu a discesa sulla sinistra dello schermo. Quando create un nuovo database, quest'ultimo verrà aggiunto a questo menu in modo da consentire un accesso rapido.

Eliminazione di database e tabelle

Una volta selezionato un database, è possibile modificare le tabelle che lo costituiscono. È anche possibile eliminare completamente una tabella. A tal fine selezionate il database dall'elenco a discesa sulla sinistra della schermata di PHPMyAdmin; dovete fare clic sul nome del database sulla sinistra della schermata sotto l'elenco a discesa. Verranno visualizzate le tabelle disponibili nel database, come illustrato nella Figura 34.12. Potete eliminare qualunque tabella del database facendo clic sulla casella accanto al nome della tabella e selezionando *Elimina* dal menu a discesa *Se selezionati*. Infine fate clic su *Esegui* per togliere la tabella dal database.

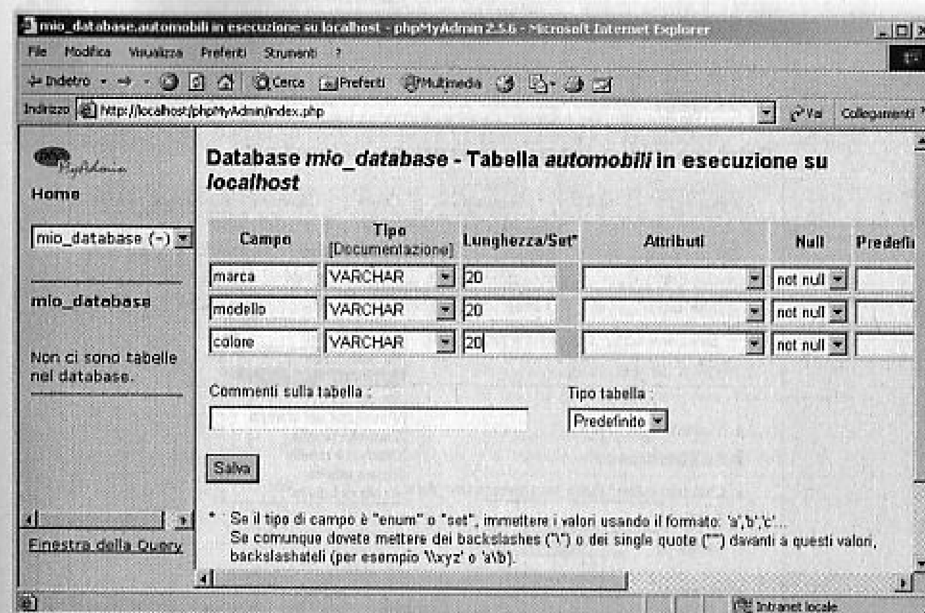


Figura 34.10
Campi della tabella completati.

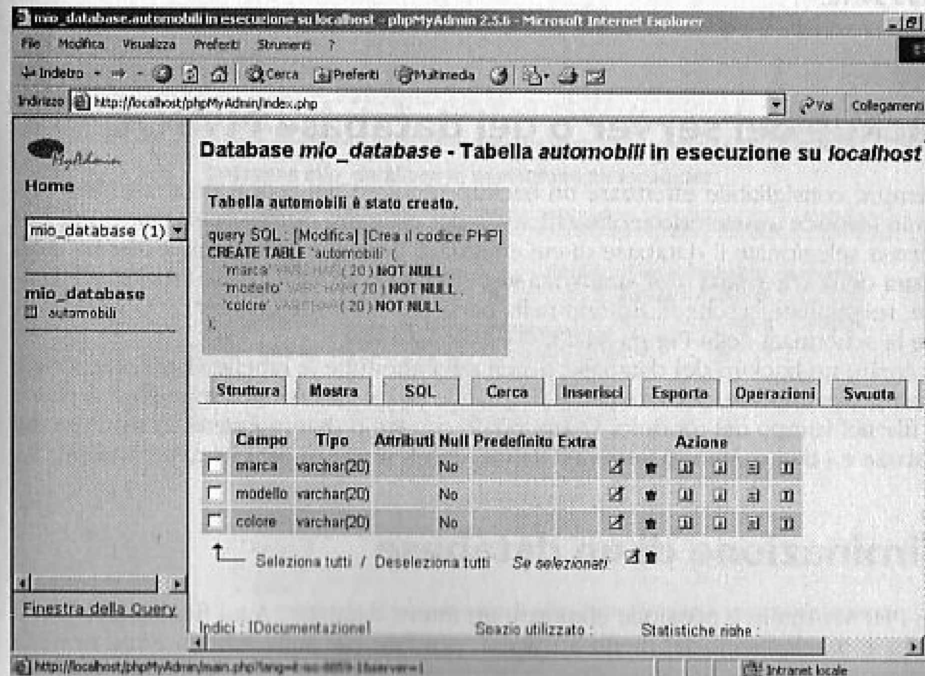


Figura 34.11
Il database "mio_database" con la tabella automobili.

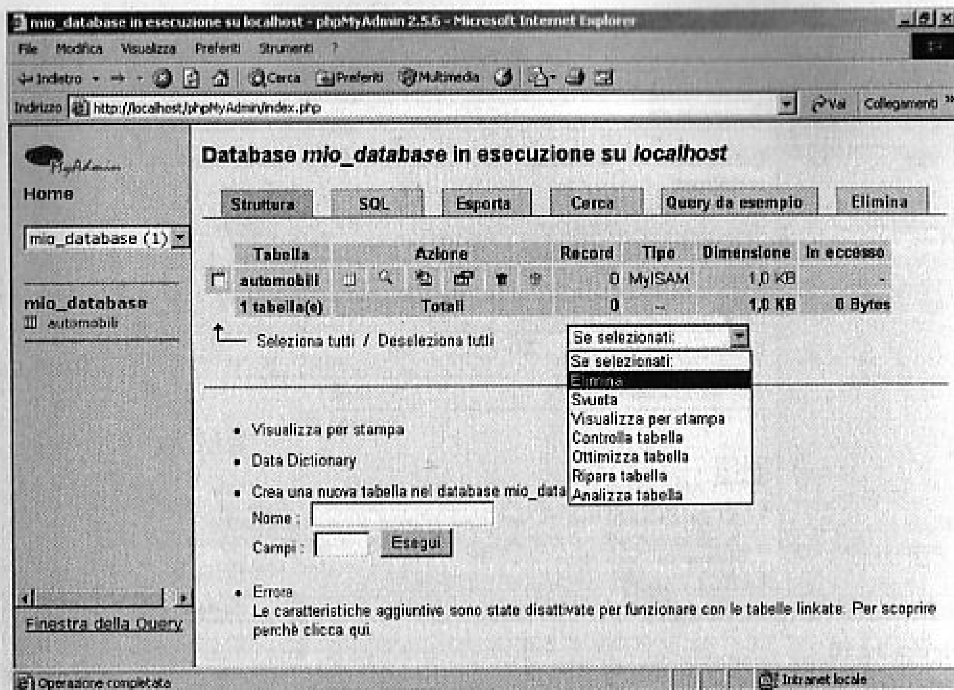


Figura 34.12
Eliminazione di tabelle o di interi database.

Backup del server o dei database MySQL

È sempre consigliabile effettuare un backup regolare dei propri database. PHPMyAdmin fornisce un metodo molto efficiente per eseguire questa operazione. Per prima cosa selezionate il database di cui effettuare il backup dal menu a discesa sulla sinistra della schermata. Poi, una volta visualizzata la struttura delle tabelle del database, selezionate la scheda *Esporta* nella parte superiore della finestra. Dovreste vedere la schermata della Figura 34.13.
Per creare un backup del database, accertatevi che tutte le tabelle siano selezionate nella finestra *Esporta*, poi fate clic sull'opzione *Salva con nome* e immettete il nome del file nel campo del modulo. Quando fate clic sul pulsante *Esegui*, la struttura del database e i dati contenuti verranno salvati con il nome di file che avete inserito.

Eliminazione di un database

Con PHPMyAdmin è possibile eliminare un intero database. A tal fine selezionate il database da eliminare dal menu a discesa, poi fate clic sulla scheda *Elimina* nella parte superiore della finestra, come mostrato nella Figura 34.14. Apparirà una finestra che chiede di confermare l'eliminazione del database.

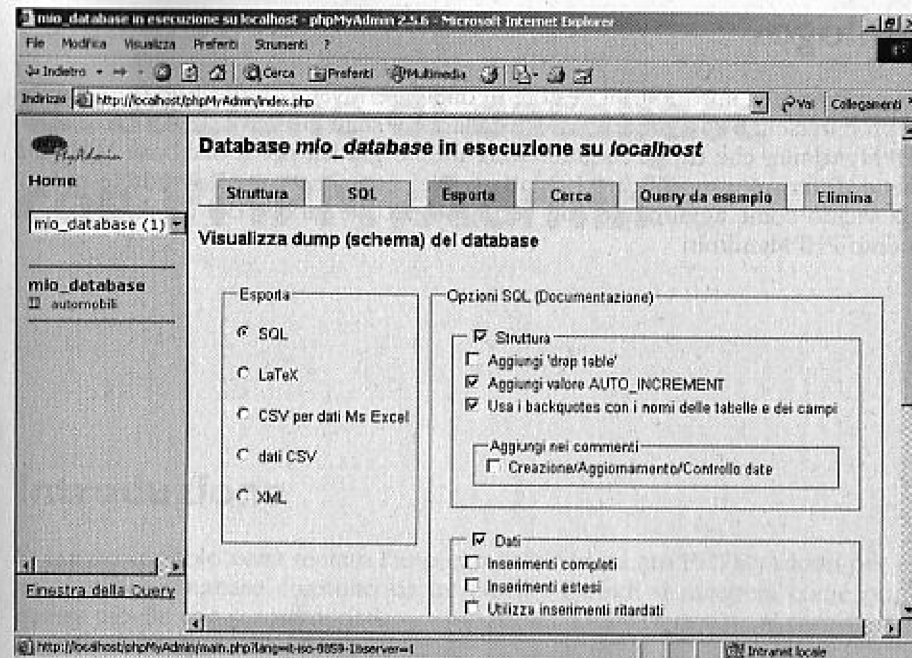


Figura 34.13
Creazione di un dump per "mio_database" (backup).

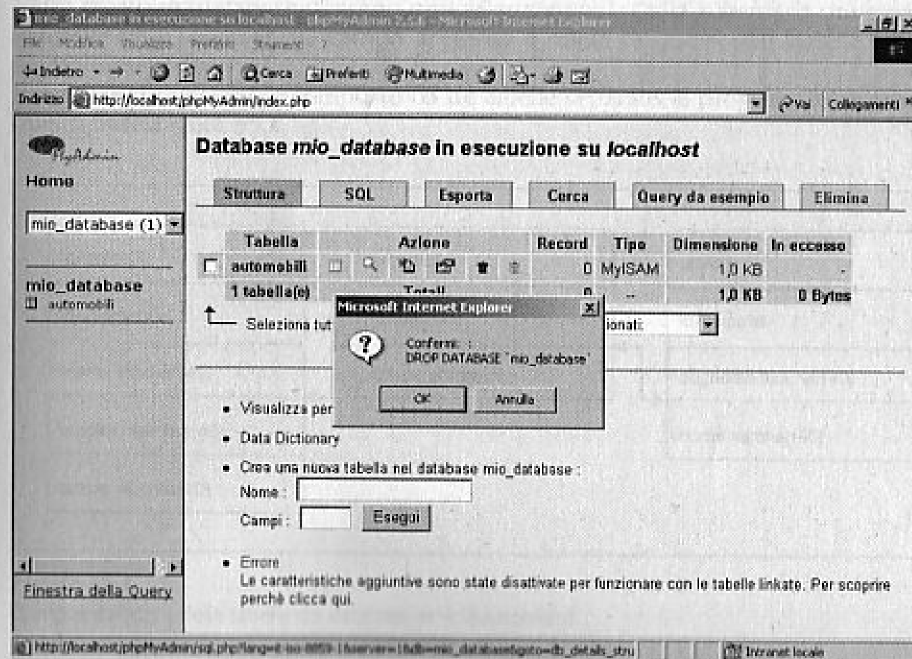


Figura 34.14
Eliminazione di un database.

Riepilogo

Questo capitolo ha introdotto il gestore di database MySQL, indicando dove è possibile procurarselo e spiegando come installarlo. È stato anche descritto lo strumento PHPMyAdmin, che funge da interfaccia utente grafica per il database MySQL e permette di creare e modificare i database. Nel prossimo capitolo vedrete un po' più da vicino come aggiungere dati veri e propri alle tabelle del database con lo strumento PHPMyAdmin.

Capitolo 35

Creazione di database con PHPMyAdmin

Introduzione

In questo capitolo verrà tentato l'impiego dello strumento PHPMyAdmin per creare un semplice database costituito da tre tabelle. Quindi si mostrerà come popolare queste tabelle con record di dati.

Creazione del database

Verrà creato un database di nome *amici&automobili*. Dalla schermata principale di PHPMyAdmin, digitate nel campo apposito il nome di questo database e fate clic sul pulsante *Crea*.

Il database dovrà essere composto da tre tabelle separate. Il progetto del database è quello della Figura 35.1.

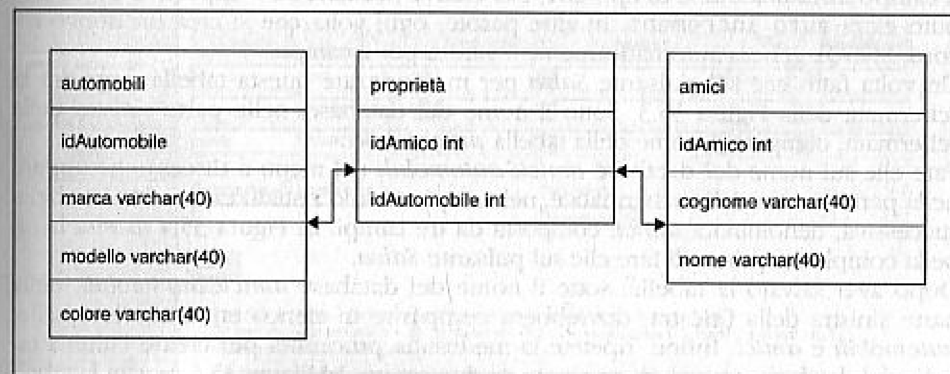


Figura 35.1
Campi e struttura delle tabelle del database amici&automobili.

Create una tabella denominata *automobili* e inseritevi quattro campi, come mostrato nella Figura 35.2.

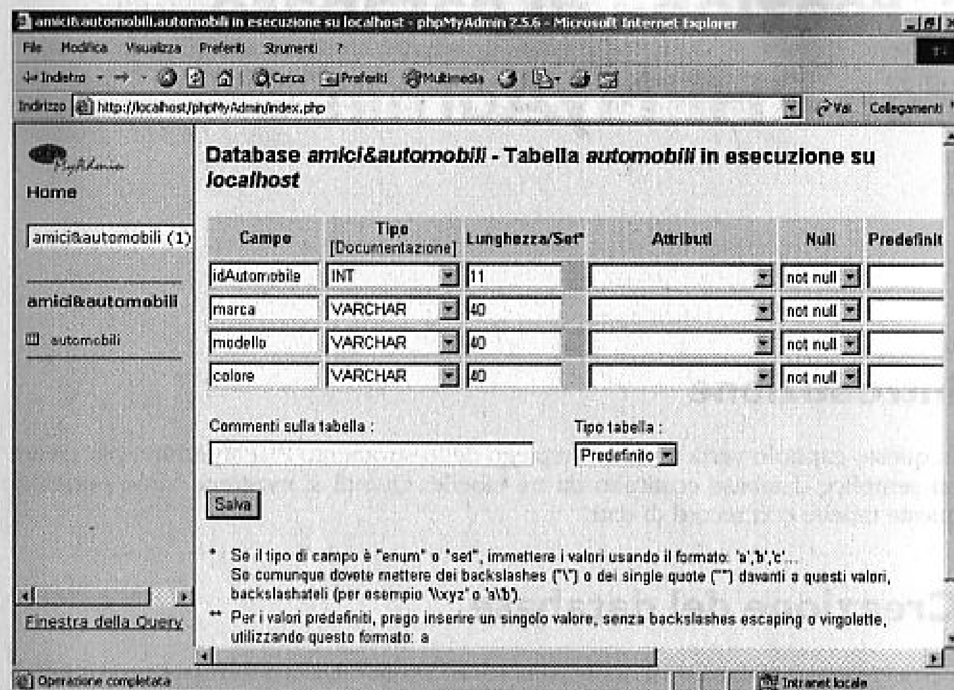


Figura 35.2
Creazione della tabella automobili e valori dei campi.

Il campo *idAutomobile* è di tipo `int`, è la chiave primaria ed è impostato con l'attributo extra `auto_increment`. In altre parole, ogni volta che si crea un nuovo record, MySQL genera automaticamente il valore per il campo.
Un volta fatto clic sul pulsante *Salva* per memorizzare questa tabella, apparirà la schermata della Figura 35.3. Sotto il nome del database, nella parte sinistra della schermata, compare il nome della tabella *automobili*.
Fate clic sul nome del database *amici&automobili* nel menu a discesa che appare nella parte sinistra della schermata e, nei campi modulo visualizzati, create la tabella successiva, denominata *amici*, composta da tre campi. La Figura 35.4 mostra la tabella completata prima di fare clic sul pulsante *Salva*.
Dopo aver salvato la tabella, sotto il nome del database *amici&automobili*, nella parte sinistra della finestra, dovrebbero comparire in elenco entrambe le tabelle: *automobili* e *amici*. Infine, ripetete la medesima procedura per creare l'ultima tabella del database, *proprietà*, costituita da due campi. La Figura 35.5 mostra la tabella completata prima di fare clic sul pulsante *Salva*.
Ora, avete completato la struttura del database e siete pronti per cominciare ad aggiungervi record di dati.

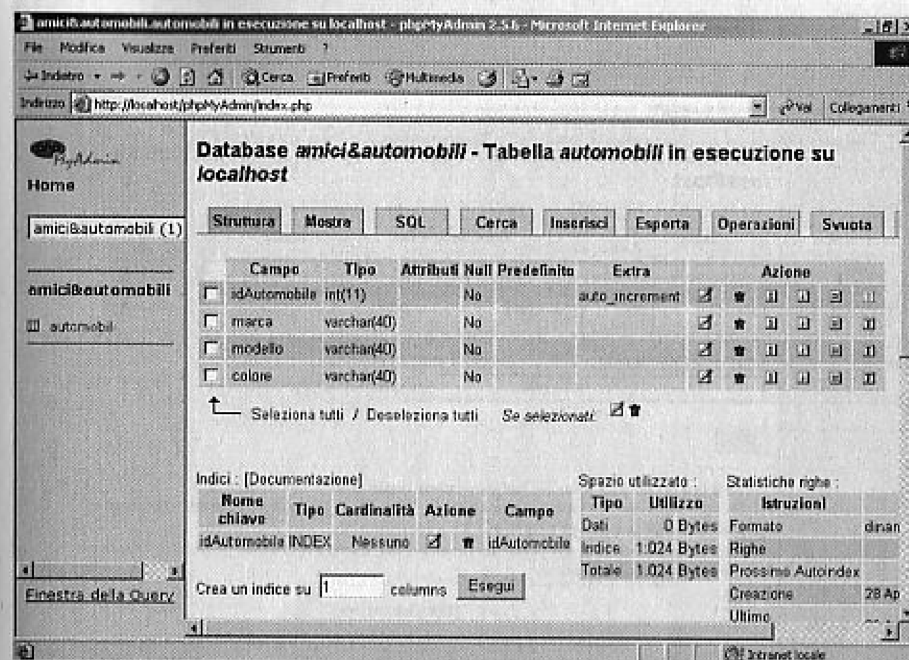


Figura 35.3
Il database amici&automobili con la tabella automobili.

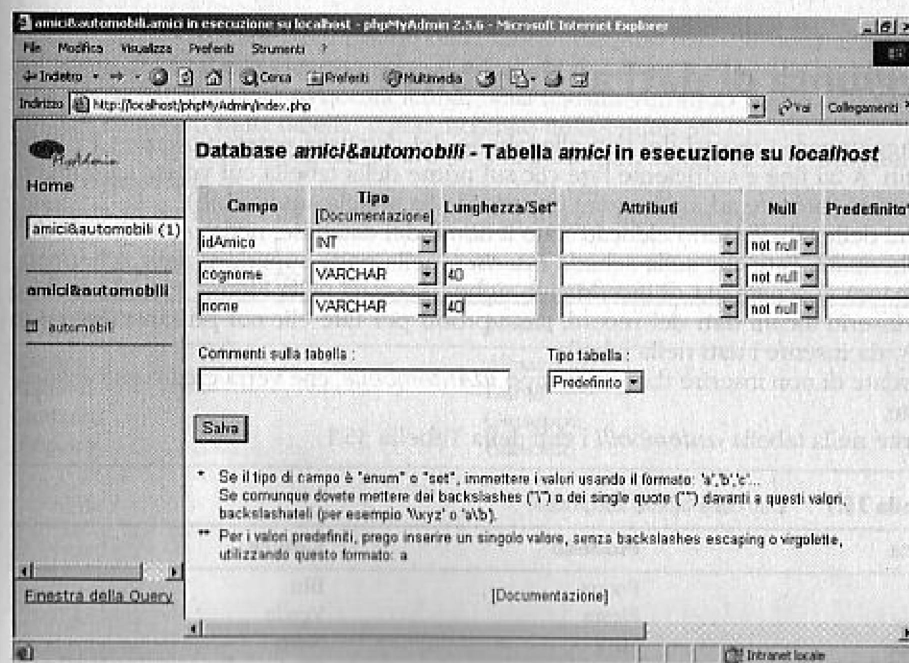


Figura 35.4
Il database amici&automobili con la tabella amici.

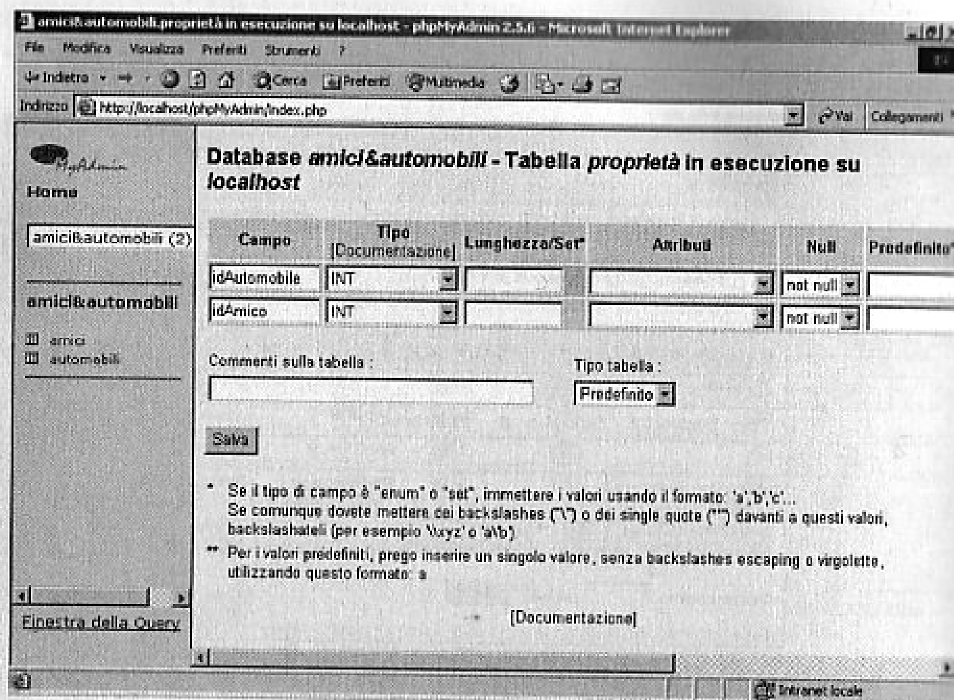


Figura 35.5
Il database amici&automobili con la tabella proprietà.

Aggiunta di dati al database

Per aggiungere i record dei dati alle tabelle del database potete utilizzare PHPMyAdmin. A tal fine è sufficiente fare clic sul nome della tabella cui volete aggiungere un record (provate ad aggiungere un record alla tabella automobili).
I nomi delle tabelle sono elencati sotto il nome del database, nella parte sinistra della schermata. Fate clic sulla scheda *Inserisci*, nella parte superiore della schermata; comparirà la schermata di inserimento righe, mostrata nella Figura 35.6, dove sono stati inseriti alcuni dati del record: siete pronti per fare clic sul pulsante *Esegui* in modo da inserire i dati nella tabella.
Ricordate di non inserire dati nel campo *idAutomobile*, che verrà creato automaticamente.
Inserite nella tabella *automobili* i dati della Tabella 35.1.

Tabella 35.1 Dati della tabella automobili

Marca	Modello	Colore
Ford	Focus	Blu
Ford	Fiesta	Verde
Mazda	MPV	Nero
Renault	Clio	Blu
Toyota	Corolla	Argento

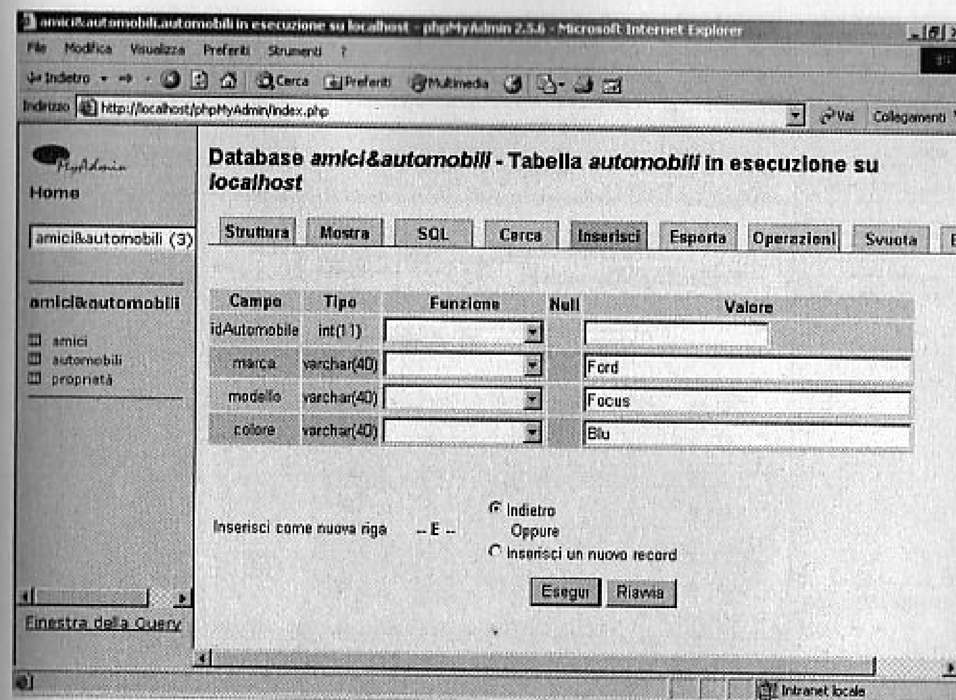


Figura 35.6
Inserimento di una riga nella tabella automobili.

È possibile visualizzare i dati inseriti nella tabella facendo clic sull'icona alla sinistra del nome della tabella elencato sotto il nome del database, nella parte sinistra della schermata. Facendo clic su questa icona, nella finestra verranno visualizzati i record dei dati contenuti nella tabella, come mostrato nella Figura 35.7.
Aggiungete i dati delle Tabelle 35.2 e 35.3 rispettivamente alle tabelle del database *amici* e *proprietà*.
La Figura 35.8 mostra i dati della Tabella 35.2 inseriti nel database.

Tabella 35.2 Dati della tabella amici

Cognome	Nome
Manzoni	Alessandro
Alighieri	Dante
Garibaldi	Giuseppe
Leopardi	Giacomo

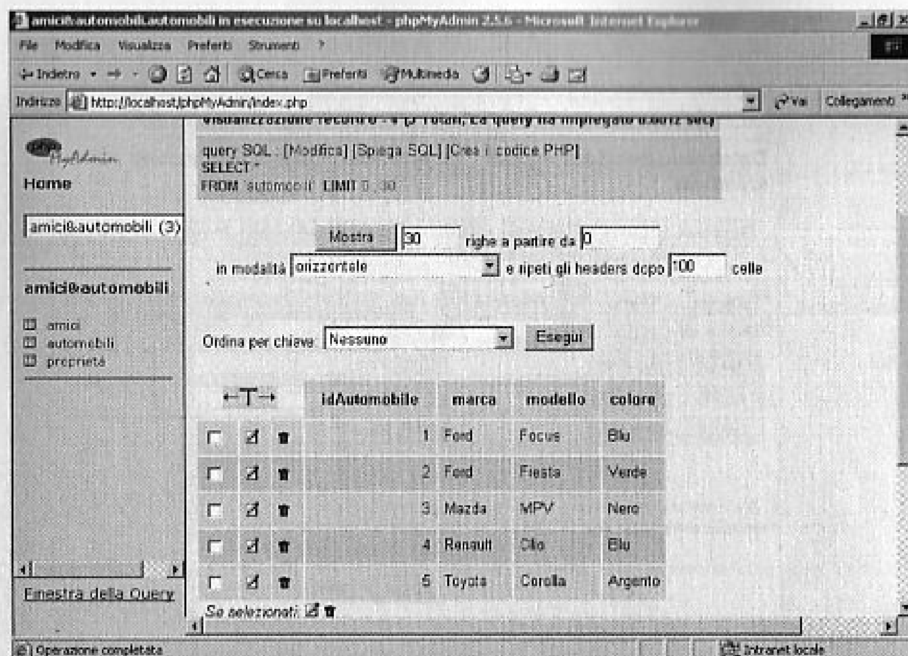


Figura 35.7

Dati della tabella automobili.

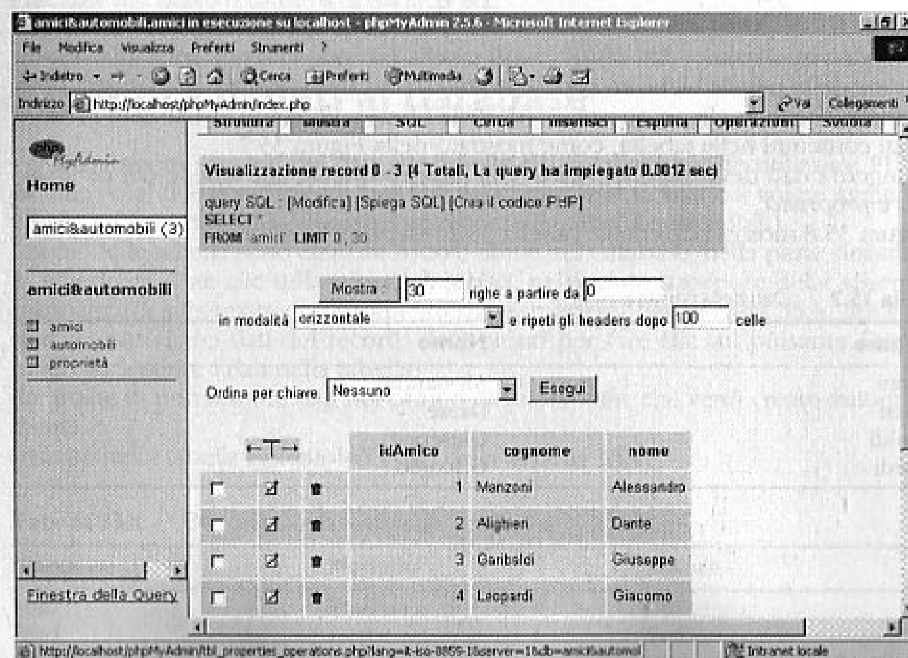


Figura 35.8

Dati della tabella amici.

La Figura 35.9 mostra i dati della Tabella 35.3 inseriti nel database.

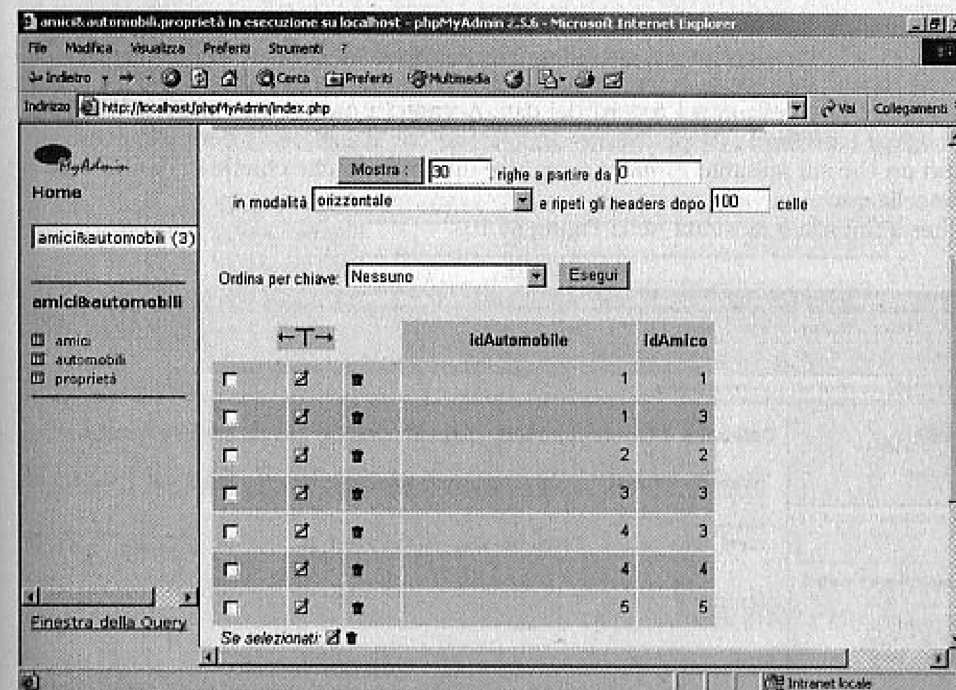


Figura 35.9

Dati della tabella proprietà.

Tabella 35.3 Dati della tabella proprietà

idAmico	idAutomobile
1	1
1	3
2	2
3	3
4	3
4	4
5	5

La tabella *proprietà* consente di specificare se un amico possiede più di un'automobile, per esempio gli amici con i numeri 1 e 4. Analogamente, potete specificare se una determinata automobile può essere posseduta da più amici; per esempio, l'automobile 3 appartiene a tre persone diverse.

Modifica e cancellazione dei dati

Se durante l'aggiunta dei record al database vi accorgete di aver commesso un errore e dovete apportare una correzione, non preoccupatevi. Quando volete cancellare un record, fate semplicemente clic sull'icona accanto al nome della tabella per visualizzare la tabella con i record dei dati. Accanto a ogni record ci sono i pulsanti *Modifica* ed *Elimina* (rispettivamente, le icone con il foglio e la penna e il cestino). Con un clic sul pulsante *Elimina*, compare una finestra che chiede di confermare la cancellazione.
Questa finestra è mostrata nella Figura 35.10.

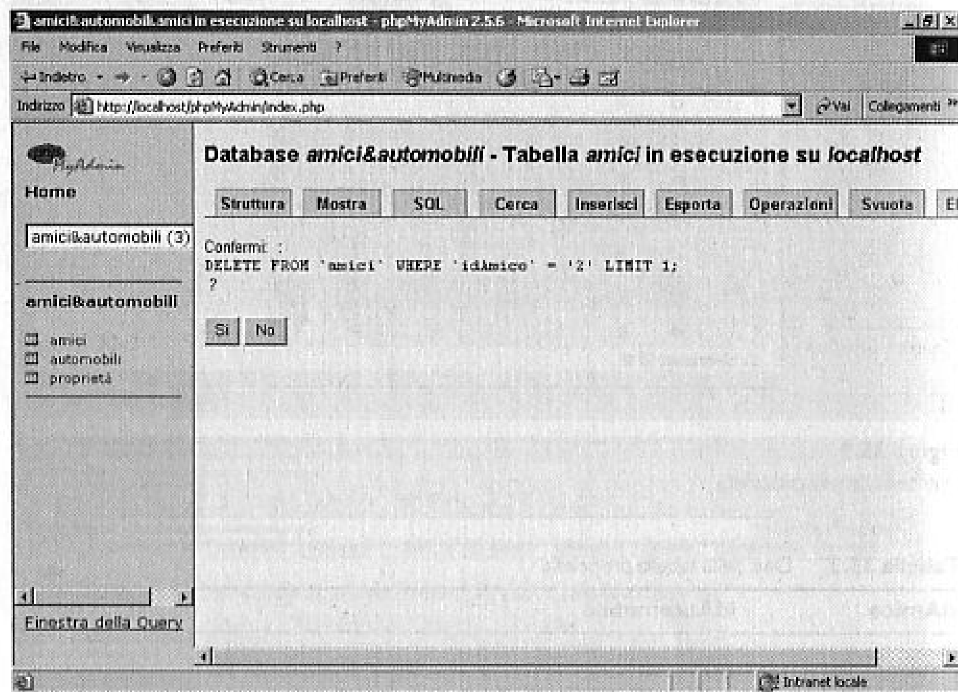


Figura 35.10
Eliminazione di dati.

Con un clic sul pulsante *Modifica*, comparirà una finestra di modifica come quella della Figura 35.11.

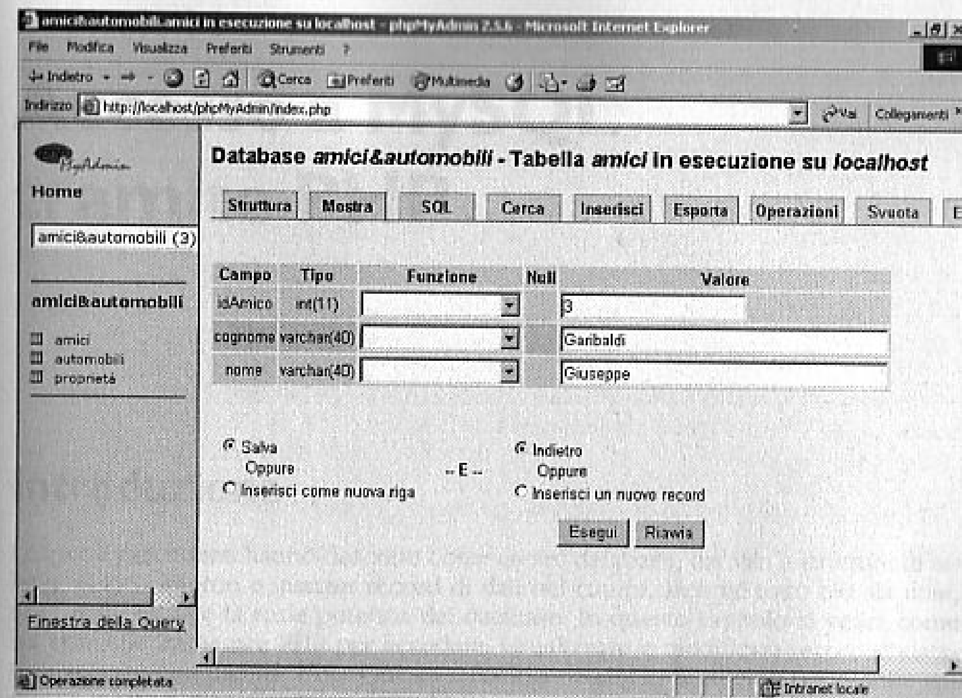


Figura 35.11
Modifica di dati.

Riepilogo

Questo capitolo ha dimostrato come utilizzare PHPMySQL per popolare un database con i dati e come manipolare questi ultimi. Tuttavia, le reali potenzialità del database MySQL si manifestano appieno solo quando si utilizza PHP per accedere ai dati. Il prossimo capitolo metterà in luce le reali capacità di MySQL collegando script PHP al database MySQL.

Capitolo 36

Accesso a MySQL tramite PHP

Introduzione

I capitoli precedenti hanno descritto come creare database, dar vita a strutture di tabelle al loro interno e inserire record di dati nei campi. Benché tutto ciò sia utile, non mette in luce la reale potenza dei database. In questo capitolo si vedrà come sia possibile impiegare PHP per accedere, visualizzare e modificare record di database tramite una pagina Web. Questa tecnologia è alla base di gran parte delle applicazioni Web dinamiche, in particolare i sistemi di e-commerce.

Connessione a un database

Per connettere uno script PHP a un database MySQL, è necessario introdurre quattro funzioni, la prima delle quali è `mysql_connect()`, che instaura una connessione dallo script alla posizione di un database specifico.
La sintassi della funzione è la seguente.

```
resource mysql_connect(string server, string nomeutente, string password);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
server	`string`	Nome del server.
nomeutente	`string`	Nome utente.
password	`string`	Password.
Restituzione di `mysql_connect()`	`resource`	Un identificatore di risorsa che punta al server.

La funzione `mysql_connect()` restituisce un tipo `resource`. Questo tipo di variabile è stato nominato per la prima volta nel Capitolo 6. Questo identificatore di risorsa risulta necessario in quanto d'ora in avanti sarà utilizzato in tutte le funzioni correlate al database.
La seconda funzione è `mysql_select_db()`, che consente di specificare quale database nella posizione definita nella funzione `mysql_connect()` desiderate utilizzare.

La sintassi della funzione è la seguente.

```
bool mysql_select_db(string nomedatabase resource identificatore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
nomedatabase	string	Nome del database.
identificatore	resource	L'identificatore di risorsa restituito dalla funzione `mysql_connect()`.
Restituzione di `mysql_select_db()`	boolean	Restituisce `TRUE` in caso di successo, `FALSE` in caso contrario.

Quindi è necessario nominare la funzione `die()`, che è un alias di `exit()`. Di questa funzione si è già parlato brevemente nel Capitolo 24. La funzione `die()` viene combinata con un operatore OR per interrompere l'esecuzione dello script nel caso in cui non sia stato possibile stabilire la connessione al database precedente. Infine, la funzione `mysql_error()` restituisce il testo del messaggio di errore dell'operazione MYSQL precedente.
La sintassi della funzione è la seguente.

```
string mysql_error()
```

La tabella seguente descrive il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
Restituzione di `mysql_error()`	string	Messaggio di errore.

Il frammento che segue offre un esempio di come queste funzioni vengano utilizzate insieme:

```
$db = mysql_connect('localhost", "root", "stobbie") or die("Impossibile
                    connettersi: " . mysql_error());
mysql_select_db("friends&cars",$db);
```

Osservate che in questo esempio di `mysql_connect()` l'host specificato è `"localhost"`; tuttavia, se il server MySQL non si trovasse sulla stessa macchina sulla quale avviene lo sviluppo del sito, questo avrebbe l'indirizzo Web di quel computer. Il secondo parametro è il nome utente, in questo caso `"root"`. Il nome utente dev'essere specificato se il database è stato creato con limitazione di sicurezza. L'ultimo parametro è la password selezionata per il proprio nome utente.
La funzione `mysql_select_db()` seleziona il database *amici&automobili* creato nel capitolo precedente.
Per agevolarne l'utilizzo sarà creato uno script MySQL esempio36-db.php nel quale viene effettuata la connessione al database.

```
<?php

//Accesso a MySQL - Esempio 36-db
//-------------------------------
$db = mysql_connect("localhost", "root", 'stobbie") or die("Impossibile
                    connettersi: " . mysql_error());
mysql_select_db('amici&automobili",$db);

?>
```

Questo file verrà incluso in tutti gli esempi di questo capitolo.

Selezione dei dati

La funzione `mysql_query()` viene impiegata per selezionare dati da un database con una query SQL (*Structured Query Language*).
La sintassi della funzione è la seguente.

```
resource mysql_query (string query, resource identificatore);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
query	string	Query SQL.
identificatore	resource	L'identificatore di risorsa restituito dalla funzione `mysql_connect()`.
Restituzione di `mysql_query()`	resource	Identificatore di risorsa per i record restituiti.

Esempio di funzione:

```
$result = mysql_query("SELECT * FROM users",$db);
```

Una volta ottenuto un insieme di risultati, è necessaria una funzione che restituisca il contenuto di una cella di record dall'insieme dei record. Questa funzione prende il nome di `mysql_result()`.
La sintassi della funzione è la seguente.

```
mixed mysql_result(resource risultato, int riga, mixed campo);
```

La tabella seguente descrive gli argomenti e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
risultato	resource	L'identificatore di risorsa restituito dalla funzione `mysql_result()`.
identificatore	int	Numero di cella da restituire, dove 0 è la prima cella.
campo	mixed	Il nome del campo da restituire.
Restituzione di `mysql_query()`	mixed	Cella del campo del database.

Esempio di funzione:

```
$rec= mysql_result($result,0, 'cognome');
```

Si comincerà con la creazione di uno script che estrae un singolo record dalla tabella *amici* del database *amici&automobili*. Considerate lo script PHP seguente:

```
<?php

//Accesso a MySQL - Esempio 36-1
//----------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>

   <?php

   $result = mysql_query("SELECT * FROM amici",$db);

   echo "nome: ". mysql_result($result,0,"nome") . "<br />";
   echo "cognome: ". mysql_result($result,0,"cognome") . "<br />";
   ?>

   </body>
<html>
```

Questo esempio visualizza il contenuto della prima riga (riga 0) del campo dei record del database. Ovviamente non si tratta ancora di un esempio molto utile, in quanto viene visualizzato un solo record. Verrà creato ora un nuovo esempio che include un ciclo in grado di visualizzare tutti i record del database. A tal fine è necessario introdurre una nuova funzione di nome `mysql_fetch_row()`.
La sintassi della funzione è la seguente.

```
array mysql_fetch_row(resource risultato);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	**Tipo**	**Descrizione**
risultato	resource	L'identificatore di risorsa restituito dalla funzione `mysql_result()`.
Restituzione di `mysql_fetch_row()`	array	Array dei record del database interrogato.

Esempio di funzione:

```
$records = mysql_fetch_row($result);
```

Dato che la funzione restituisce un array dei campi per il record, l'accesso a questi avviene con l'indicizzazione dell'array. Per esempio, la seguente istruzione restituisce il secondo campo di un record.

```
$records[1];
```

Lo script che segue illustra una versione modificata dell'esempio precedente, che visualizzerà tutti i record presenti nel database.

```
<?php

//Accesso a MySQL - Esempio 36-2
//----------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>

   <?php

   $result = mysql_query("SELECT * FROM amici",$db);

   while ($records = mysql_fetch_row($result)) {
      echo "nome: ". $records[2] . "<br />";
      echo "cognome: ". $records[1] . "<br />";
   }

   ?>s
   </body>
<html>
```

Questo script è simile all'esempio precedente, però introduce un'istruzione di ciclo:

```
while ($rec = mysql_fetch_row($result)) { }
```

Osservate che nel ciclo si trova una nuova funzione, `mysql_fetch_row()`, che restituisce un singolo record, memorizzato nella variabile `$result`. Il record viene poi memorizzato in un array di nome `$rec`. Le istruzioni `echo` visualizzano ciascun elemento dell'array `$rec[]`, che contiene il valore di ogni campo dei record. L'elemento [0] dell'array contiene l'ID del record, mentre [1] contiene il nome degli amici e [2] il loro cognome. Ciascuna iterazione del ciclo `while` determinerà la chiamata della funzione `mysql_fetch_row()`, che ogni volta restituirà il record successivo del database. Quando non trova più record, restituisce `FALSE` e il ciclo viene terminato.
L'output di questo script è quello della Figura 36.1.
Nell'esempio precedente veniva utilizzata la funzione `mysql_fetch_row()` per accedere ai record del database e visualizzarli. Benché questa soluzione funzioni, il programmatore che scrive lo script deve fare riferimento ai campi separati tramite gli indici dell'array, il che non agevola la lettura del codice e lo espone all'introduzione di errori. La funzione `mysql_fetch_array()` è una versione estesa di `mysql_fetch_row()`.
La sintassi della funzione è la seguente.

```
array mysql_fetch_array(resource risultato);
```

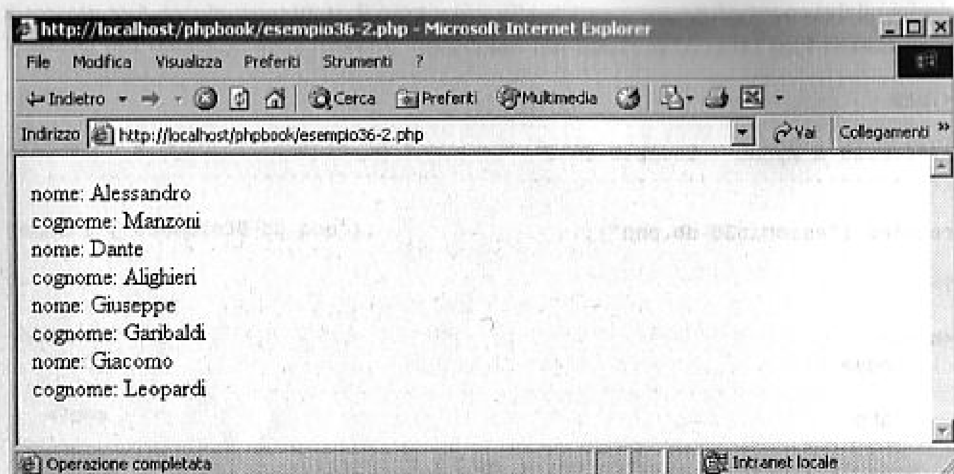


Figura 36.1
Record del database.

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
risultato	`resource`	L'identificatore di risorsa restituito dalla funzione `mysql_result()`.
Restituzione di `mysql_fetch_array()`	`array`	Array dei record del database chiamato in causa.

Esempio di funzione:

```
$records = mysql_fetch_array($result);
```

Questa funzione è uguale a `mysql_fetch_row()`, tranne per il fatto che consente di fare riferimento ai campi del database utilizzandone il nome. Lo script che segue illustra l'utilizzo di questa funzione.

```
<?php

//Accesso a MySQL - Esempio 36-3
//----------------------------------

require ("esempio36-db.php");
?>

<html>
   <body>

   <?php

   $result = mysql_query("SELECT * FROM amici",$db);

   while ($records = mysql_fetch_array($result)) {
      echo "nome: ". $records['nome'] . "<br />";
      echo "cognome: ". $records['cognome'] . "<br />";
```

```
   }

   ?>

   </body>
<html>
```

Tecniche avanzate di selezione dei dati

Le istruzioni delle query SQL che vengono inviate al database con la funzione `mysql_query()` sono in grado di eseguire operazioni molto più complesse rispetto agli esempi illustrati finora in questo capitolo. Occorre soffermarsi sull'analisi di alcune caratteristiche che possono essere di aiuto per selezionare i dati richiesti dal database.

Selezione di un certo numero di record

Le query SQL possono utilizzare l'opzione `LIMIT` per selezionare un certo numero di record da una tabella. L'opzione `LIMIT` richiede due parametri: il primo (*x*) rappresenta la riga di partenza, mentre il secondo (*y*) rappresenta il numero di record che devono essere selezionati a partire da *x*. Per esempio:

```
$query = mysql_query ("SELECT campi FROM tabella LIMIT 1, 3", $db);
```

Questo esempio seleziona 3 record a partire dalla riga 1. Lo script che segue illustra quanto appena affermato.

```
<?php

//Accesso a MySQL - Esempio 36-4
//----------------------------------

require ("esempio36-db.php");
?>

<html>
   <body>

   <?php

   $result = mysql_query("SELECT * FROM amici LIMIT 1,3",$db);

   while ($records = mysql_fetch_array($result)) {
      echo "nome: ". $records['nome'] . "<br />";
      echo "cognome: ". $records['cognome'] . "<br />";
   }

   ?>

   </body>
<html>
```

L'output di questo script è uguale a quello del precedente, tranne per il fatto che vengono visualizzati solo tre record.

Ordinamento dei record

Le query SQL possono impiegare l'opzione `ORDER BY` per ordinare i dati selezionati. L'opzione `ORDER BY` dev'essere seguita dal nome della colonna (campo) e da un'opzione aggiuntiva che definisce se l'ordinamento dev'essere crescente (`asc`) o decrescente (`desc`):

```
$query = mysql_query ("SELECT field(s) FROM table ORDER BY cognome desc", $db);
```

Lo script che segue mostra i risultati ordinati in ordine crescente di cognome:

```
<?php

//Accesso a MySQL - Esempio 36-5
//-------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>
   <?php

   $result = mysql_query("SELECT * FROM amici ORDER BY cognome asc",$db);

   while ($records = mysql_fetch_array($result)) {
      echo "nome: ". $records['nome'] . "<br />";
      echo "cognome: ". $records['cognome'] . "<br />";
   }

   ?>

   </body>
<html>
```

L'output di questo script è quello della Figura 36.2.

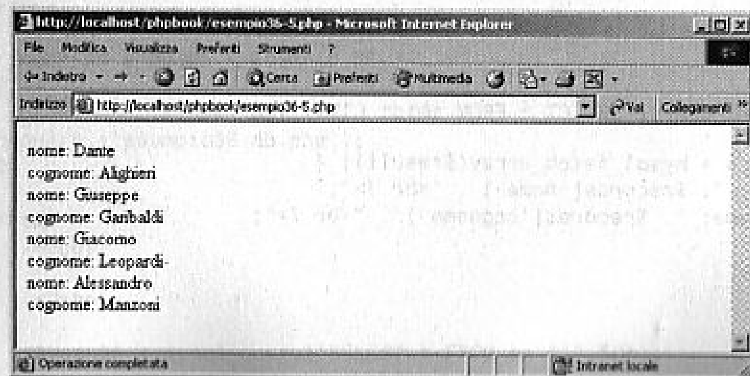


Figura 36.2
Output ordinato per cognome.

Inserimento di dati

Finora avete visto come estrarre dati da un database, tuttavia sarebbe utile anche poterne aggiungere di nuovi. La funzione `mysql_query()` utilizzata con la query `INSERT INTO` è la soluzione a questo problema:

```
$insert = mysql_query("INSERT INTO tabella (colonna1, colonna2, ...,
colonna-x) VALUES ('valore', 'valore', ..., 'valore')" ,$db);
```

Con una query `INSERT INTO`, i nomi dei campi della tabella (ossia delle colonne) devono essere specificati nello stesso ordine dei valori di questi ultimi. Non c'è alcuna restrizione al numero di colonne utilizzabili per l'inserimento di dati, tuttavia si consiglia di specificare tutte le colonne in una query `INSERT INTO` indicando valori vuoti per quelle che desiderate lasciare vuote.
Potete comunque evitare di utilizzare le colonne e ricorrere a una struttura alternativa per la query di inserimento, come nell'esempio che segue. Osservate che il primo valore verrà inserito nella prima colonna, il secondo nella seconda e così via.

```
$insert = mysql_query("INSERT INTO tabella VALUES ('valore', 'valore',
..., 'valore')" ,$db);
```

L'esempio che segue utilizza la query `INSERT` per aggiungere un nuovo record alla tabella *amici*.

```
<?php

//Accesso a MySQL - Esempio 36-6
//-------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>

   <?php

   $insert = mysql_query("INSERT INTO amici VALUES ('', 'Foscolo', 'Ugo')"
                     ,$db);

   $result = mysql_query("SELECT * FROM amici ORDER BY cognome asc", $db);

   while ($records = mysql_fetch_array($result)) {
      echo "Id: ". $records['idAmico'] . "<br />";
      echo "nome: ". $records['nome'] . "<br />";
      echo "cognome: ". $records['cognome'] . "<br /><br />";
   }

   ?>

   </body>
<html>
```

L'output di questo script è quello della Figura 36.3 e mostra il record appena aggiunto.

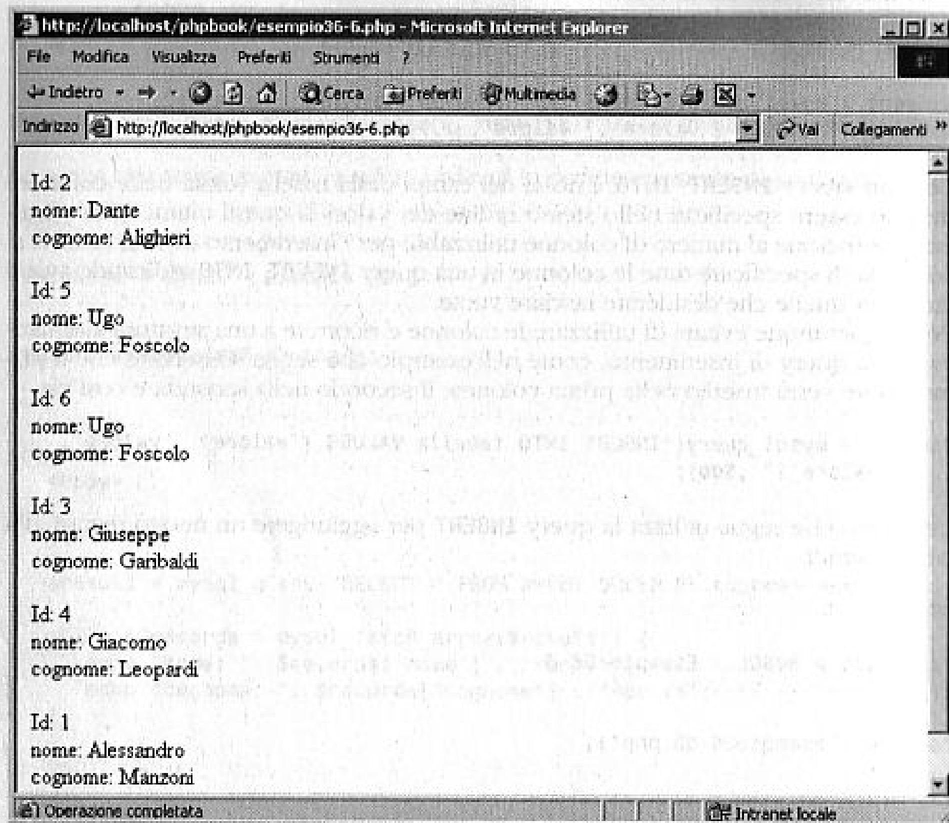


Figura 36.3
Inserimento di record.

Eliminazione di dati

È possibile eliminare record dalle tabelle utilizzando la funzione `mysql_query()` con la query `DELETE FROM`:

```
$DELETE = mysql_query ("DELETE FROM tabella WHERE colonna='$valore'",
$db);
```

L'esempio che segue illustra uno script che consentirà di eliminare qualunque record sia stato inserito nel database a patto di immetterne il numero di ID.

```
<?php

//Accesso a MySQL - Esempio 36-7
//-------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>
      <?php

      if (isset($_POST['id'])) {
         $delete = mysql_query("DELETE FROM amici WHERE
         idAmico=$_POST[id]",$db);
         echo ("<br>Record eliminato<br>");
      }

      $result = mysql_query("SELECT * FROM amici",$db);

      while ($records = mysql_fetch_array($result)) {
         echo "Id: ". $records['idAmico'] . "<br />";
         echo "nome: ". $records['nome'] . "<br />";
         echo "cognome: ". $records['cognome'] . "<br /><br />";
      }

      ?>
      <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">

      Id: <input type="text" name="id"><br>
      <input type="submit" value="Elimina record">

      </form>
   </body>
<html>
```

L'output di questo script è quello della Figura 36.4, dove il record con id 5 è appena stato eliminato.
Questo script contiene un'istruzione `if` che verifica il valore di `$id`, il quale indica se il modulo è stato inviato. `$id` viene utilizzata per selezionare il record da eliminare:

```
      if (isset($_POST['id'])) {
```

Se la variabile `$id` contiene un valore, la funzione `mysql_query()` viene chiamata:

```
         $delete = mysql_query("DELETE FROM amici WHERE
         idAmico=$_POST[id]",$db);
         echo ("<br>Record eliminato<br>");
      }
```

Questa funzione elimina il record nel database che ha un id pari a `$id`. Osservate che questo script non contiene alcuna forma di controllo di errore, quindi l'utente è libero di digitare un numero di id qualunque.

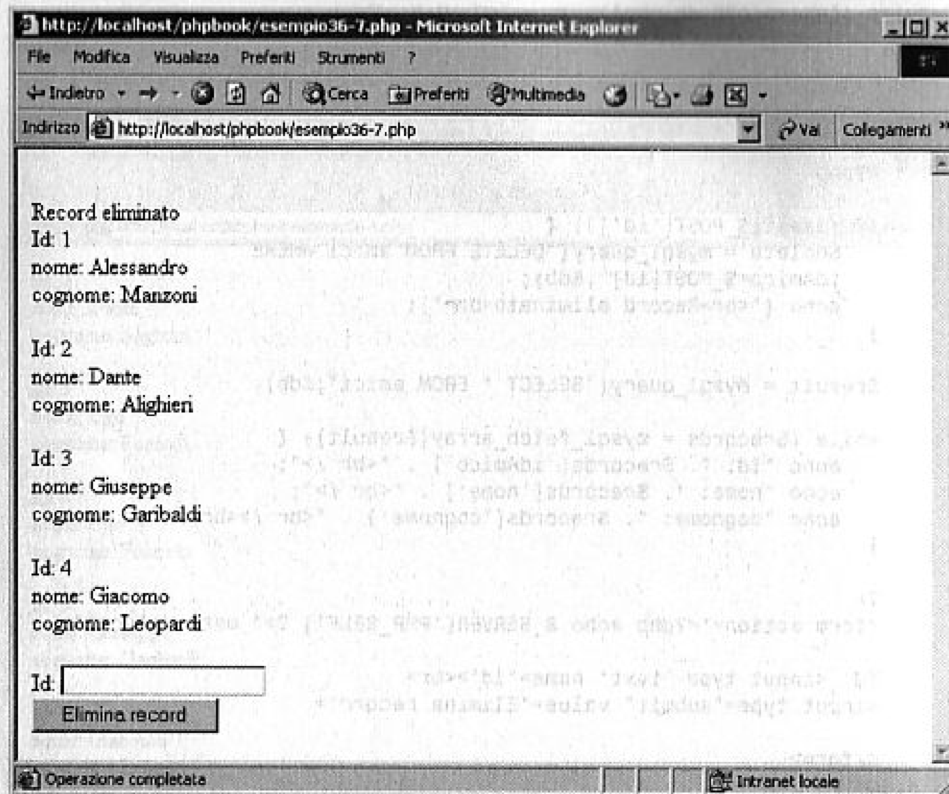


Figura 36.4
Il record con id 5 è stato eliminato.

Aggiornamento dei dati

la funzione `mysql_query()` può essere utilizzata per consentire l'aggiornamento di un record esistente. Ciò avviene con il ricorso alla query `UPDATE`.

```
$update = mysql_query ("UPDATE tabella SET colonna1='valore',
colonna2='valore' ... colonnaX='valore' WHERE colonna='valore'",$db);
```

Lo script che segue utilizza un modulo per aggiornare i record esistenti. Per garantire che venga modificato il record corretto, si ricorre al suo id.

```
<?php

//Accesso a MySQL - Esempio 36-8
//--------------------------------

require ("esempio36-db.php");

?>
```



```
<html>
   <body>
      <?php

      if (isset($_POST['id'])) {

         $result = mysql_query("UPDATE amici
            SET cognome='$_POST[cognome]',
            nome='$_POST[nome]'
            WHERE idAmico='$_POST[id]'",$db);
      }

      $result = mysql_query("SELECT * FROM amici",$db);

      while ($records = mysql_fetch_array($result)) {
         echo "Id: ". $records['idAmico'] . "<br />";
         echo "nome: ". $records['nome'] . "<br />";
         echo "cognome: ". $records['cognome'] . "<br /><br />";
      }

      ?>

      <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">

      Id: <input type="text" name="id"><br />
      Nome: <input type="text" name="nome"><br />
      Cognome: <input type="text" name="cognome"><br />
      <input type="submit" value="Aggiorna record">

      </form>
   </body>
<html>
```

Conteggio delle righe e verifica dell'esistenza dei record

La funzione `mysql_num_rows()` può essere utilizzata per contare il numero dei record in una tabella e verificare se un record esiste.
La sintassi della funzione è la seguente.

```
int mysql_num_rows(resource risultato);
```

La tabella seguente descrive l'argomento e il tipo di valore restituito dalla funzione.

Nome	Tipo	Descrizione
`risultato`	`resource`	L'identificatore di risorsa restituito dalla funzione `mysql_result()`.
Restituzione di `mysql_num_rows()`	`int`	Numero di righe di record.

Esempio di funzione:

```
$records = mysql_num_rows($result);
```

Considerate l'esempio che segue, dove la funzione `mysql_num_rows()` viene utilizzata per contare i record (le righe) correnti della tabella *amici*.

```
<?php

//Accesso a MySQL - Esempio 36-9
//--------------------------------

require ("esempio36-db.php");

?>

<html>
   <body>

   <?php
   $result = mysql_query("SELECT * FROM amici",$db);

   $rows = mysql_num_rows($result);

   if($rows) {
      echo 'Trovati $rows record';
   }
   else {
      echo 'Nessun record trovato';
   }

   ?>

   </body>
<html>
```

Questo script visualizzerà il messaggio "Trovati 4 record".

Riepilogo

Questo capitolo ha mostrato come sia possibile utilizzare uno script PHP per estrarre dati da un database MySQL. Inoltre si è visto che PHP è anche in grado di eliminare e correggere i record esistenti. Argomento del prossimo capitolo saranno i concetti di classe e oggetto, nucleo fondamentale della programmazione a oggetti.

Parte XI

Concetti di classe e oggetto

Capitolo 37

Classi e oggetti

Introduzione

Molti linguaggi di programmazione moderni supportano il paradigma dell'orientamento agli oggetti. Si tratta semplicemente di un modo alternativo di progettare e implementare i programmi. Nell'ambito del paradigma orientato agli oggetti, i programmatori modellano elementi noti come classi e li utilizzano all'interno di un programma creando istanze di queste classi, note con il nome di "oggetti". PHP supporta la creazione di classi e l'implementazione di oggetti e permette al programmatore di sviluppare programmi orientati agli oggetti. Questo capitolo spiegherà il concetto dell'orientamento agli oggetti e fornirà qualche esempio degli elementi di tale paradigma supportati da PHP.

Che cosa sono le classi e gli oggetti

L'orientamento agli oggetti è un paradigma che incoraggia e aiuta a riutilizzare il codice. Inoltre tende a ridurre al minimo l'impatto di qualsiasi modifica di programmazione, grazie a una tecnica nota con il nome di "incapsulamento". L'orientamento agli oggetti è un modo diverso di concepire la soluzione a un problema di programmazione. Le tecniche di programmazione strutturata tradizionali risolvono i problemi scomponendo la soluzione in funzioni facili da programmare. Queste funzioni vengono raggruppate all'interno del programma per eseguire le operazioni necessarie (Figura 37.1). Si tratta della tecnica adottata per risolvere i problemi emersi in questo libro.
Anche se nel corso degli anni questa tecnica si è rilevata molto utile, essa non aiuta i programmatori a riutilizzare in modo efficace il codice e non contribuisce nemmeno a proteggere i dati da modifiche accidentali. Con un approccio funzionale, i dati locali possono essere dichiarati all'interno di una funzione, tuttavia potrebbero essere richiesti come parametro per un'altra funzione per consentire un'ulteriore elaborazione. Se le strutture di dati passate da una funzione all'altra sono grandi e complesse, può accadere che si verifichino errori di programmazione che possono alterare i dati.
La programmazione a oggetti comporta un approccio diverso alla progettazione dei dati. In un progetto a oggetti, i dati e i metodi (noti anche come funzioni) vengono raccolti in una classe e non è possibile accedere a essi dall'esterno; i dati risultano così protetti dal resto del programma. Si tratta della già citata tecnica dell'incapsula-

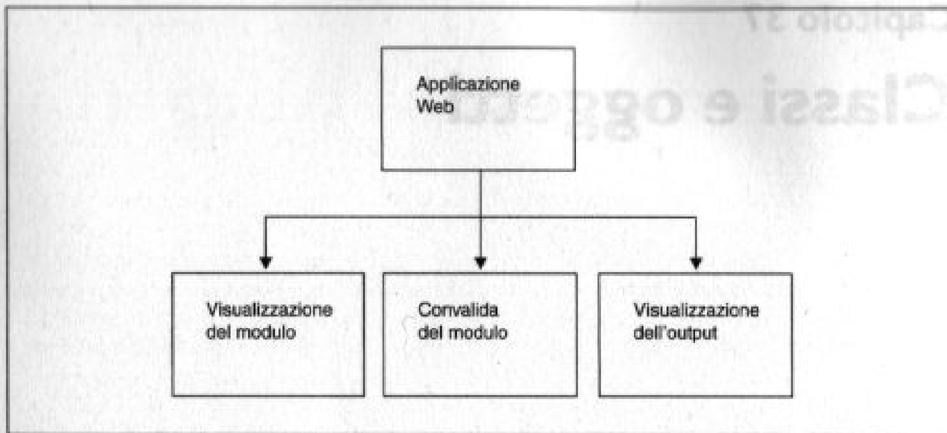


Figura 37.1
Progetto di programmazione funzionale.

mento. Solo i metodi associati alla classe sono in grado di accedere ai dati e alterarli, creando una sorta di "interfaccia" con i dati stessi.
Questa particolare invocazione dei metodi di una classe è nota come invio di un messaggio alla classe. Per vederne un esempio, fate riferimento alla Figura 37.2.

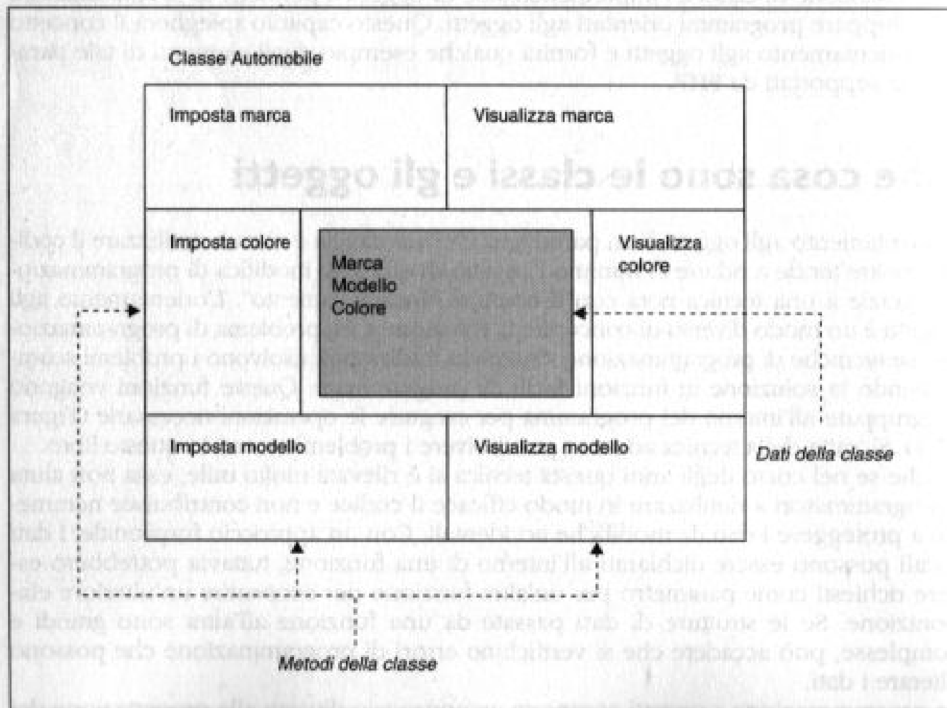


Figura 37.2
Orientamento agli oggetti: classi.

L'orientamento agli oggetti incoraggia il riutilizzo del software, in quanto consente al programmatore di definire nuove classi simili a quelle già esistenti. Il programmatore è quindi in grado di affermare: "Questa classe è identica a quell'altra, eccetto che per qualche piccola differenza". Il grande vantaggio è che il programmatore deve solo indicare le differenze, poiché tutto il codice già scritto per la classe originale diventa automaticamente accessibile alla nuova classe. Questa caratteristica è nota con il nome di "ereditarietà". L'orientamento agli oggetti supporta anche il polimorfismo, il concetto secondo cui alcune classi correlate hanno tutte un metodo con lo stesso nome. Per esempio, potreste aver creato alcune classi che disegnano forme diverse sul monitor. Tutte le forme hanno un metodo, chiamato `draw()`, che consente loro di visualizzare se stesse sullo schermo. Il programmatore può richiamare il metodo `draw()` in una qualsiasi di queste classi sapendo che sullo schermo comparirà una forma. Tuttavia l'immagine risultante cambierà in base alla classe cui viene inviato il messaggio.
Il lavoro non termina con la creazione delle classi. In realtà, infatti, le classi sono modelli che definiscono quali dati è possibile contenere e come gli stessi possano essere utilizzati. In pratica, le classi sono una versione sofisticata delle variabili. Le variabili sono progettate per contenere dati di un tipo specifico e con esse si possono eseguire determinate operazioni. Tuttavia, prima di poter utilizzare una variabile è necessario crearne un'istanza. Lo stesso vale per le classi. La classe è un modello e per utilizzarla è necessario creare un'istanza della classe che contenga dati. L'istanza di una classe prende il nome di "oggetto".

Classi e oggetti in PHP

Anche se la maggior parte dei linguaggi di programmazione a oggetti implementa le funzionalità descritte in precedenza, ciò non vale per tutti i linguaggi. Infatti, in base al linguaggio di programmazione utilizzato, la terminologia cambia. Per esempio, i termini "metodi di classe", "funzioni" e "procedure" descrivono tutti la medesima cosa. Analogamente, "membri di dati" e "variabili di classe" sono espressioni che fanno riferimento al medesimo concetto.
PHP mantiene una terminologia semplice e coerente con gli altri aspetti del linguaggio. Nella definizione di una classe, il programmatore specifica le variabili e le funzioni incluse nella classe stessa. Non saranno utilizzati nomi nuovi per questi elementi poiché ciò provocherebbe confusione. Le istanze delle classi sono indicate come oggetti. Proverete ora a creare una semplice classe per capire come sia facile lavorare con esse.

Creazione di una semplice classe

La parola chiave utilizzata per definire una classe è `class`. A ogni classe dev'essere assegnato un nome univoco e occorre utilizzare le parentesi graffe `{ }` per indicare l'inizio e la fine della definizione di una classe. Per esempio, per creare una classe denominata `veicolo` utilizzerete la sintassi seguente:

```
<?php

class veicolo {
}

?>
```

All'interno delle classi è possibile incapsulare variabili. Le variabili di classe sono definite come le altre variabili PHP; l'unica differenza riguarda la parola chiave `var`, che deve precedere la variabile stessa. Per esempio:

```
var $description;
```

Le variabili di classe devono essere inserite all'interno della definizione di classe, tra le parentesi graffe iniziale e finale. Per esempio:

```
<?php

class veicolo {
   var $description;
}

?>
```

Alle classi è anche possibile associare funzioni, che offrono un'interfaccia per i dati della classe. Tutte le funzioni contenute nella classe possono accedere a tutte le variabili della classe. Ogni funzione di classe viene definita esattamente come qualunque altra normale funzione esterna. Per esempio:

```
function display() {
}
```

Le funzioni di classe possono essere create in modo da ricevere parametri e/o restituire un valore. Per esempio:

```
function set($d) {
}
```

Anche le funzioni devono essere racchiuse tra le parentesi graffe iniziale e finale della classe:

```
<?php

class veicolo {
   var $description;

   function set($d) {
   }

   function display() {
   }
}

?>
```

Per il momento, queste funzioni non eseguono alcuna operazione, poiché non contengono codice. In questo caso si vuole che la funzione `set()` assegni la variabile `$d` indicata come parametro alla variabile di classe `$description`. Questa operazione implementa una funzione che permette di impostare il valore di `$description` quando si crea un oggetto della classe. Probabilmente state pensando che l'istruzione:

```
$description = $d;
```


assegni il valore `$d` a `$description`. Non è così. Dato che `$description` è una variabile di classe e non viene dichiarata o passata alla funzione `set()`, è necessaria una sintassi speciale per fare riferimento a `$description`. A tal fine verrà utilizzata la variabile `$this`, che significa "questa classe", e l'operatore `->` seguito dal nome della variabile. Per esempio:

```
$this->description
```

Osservate che non è stato inserito un simbolo `$` prima del nome della variabile. Per assegnare una variabile `$d` a questa variabile utilizzerete l'istruzione:

```
this->description = $d;
```

Per quanto riguarda la funzione `display()`, si vuole che visualizzi il valore di `$description`. A tal fine, utilizzerete l'istruzione:

```
echo($this->description);
```

La classe che segue include queste funzioni complete.

```
<?php

class veicolo {
   var $description;

   function set($d) {
      $this->description = $d;
   }

   function display() {
      echo($this->description);
   }
}

?>
```

Ora, disponete di una classe che contiene una variabile e due funzioni che consentono di impostare e visualizzare il valore della variabile. Se provaste a visualizzare l'output dello script precedente in un browser rimarreste alquanto delusi, poiché non comparirebbe alcunché. Infatti, è stata creata una classe in modo corretto, ma non è stata ancora utilizzata. Ora dovete creare un'istanza di questa classe, ossia un oggetto.

Utilizzo della classe veicolo

Per utilizzare la classe `veicolo`, dovete creare un oggetto (una variabile, se preferite) di questa classe. A tal fine è necessario utilizzare la parola chiave `new` con la sintassi seguente:

```
$NomeOggetto= new nomeClasse;
```

Pertanto, per creare un oggetto di nome `$vei`, che è un'istanza della classe `veicolo`, dovrete scrivere:

```
$vei = new veicolo;
```

Ciò in pratica equivale a dichiarare una variabile `$vei`, però non avete ancora assegnato alcun valore. La classe è stata progettata per contenere un solo dato nella variabile `$description`. Non potete scrivere semplicemente:

```
$description = "Un'automobile blu";
```

poiché non potete assegnare valori alle classi, ma solo agli oggetti. Allora forse è possibile scrivere:

```
$vei->description = "Un'automobile blu";
```

In questo modo cercate di assegnare un valore a una variabile specifica, appartenente a un oggetto che avete creato. Tuttavia una delle "regole" alla base della programmazione a oggetti è l'incapsulamento. Il concetto di incapsulamento prevede l'occultamento dei dati all'interno di un oggetto e non permette di visualizzarli o modificarli. In che modo potete assegnare un valore a questa variabile? Attraverso le funzioni di classe che avrete creato in precedenza.

Per assegnare un valore agli oggetti occorre invocare la funzione `set()`, che è stata progettata per ricevere un parametro e memorizzarlo nella variabile di classe `$description`. Per richiamare la funzione di un oggetto si ricorre alla sintassi seguente:

```
$NomeOggetto->nomeFunzione()
```

Pertanto, per invocare la funzione `set()` di un oggetto `$vei` passandole i dati `"La mia automobile nuova fiammante."`, dovrete scrivere:

```
$vei->set("La mia automobile nuova fiammante.");
```

Questa istruzione passa la stringa `"La mia automobile nuova fiammante."` alla funzione `set()`, che appartiene agli oggetti `vei`. La funzione memorizza la stringa nella variabile di classe `$description`. Per visualizzare il valore memorizzato nella stringa, è necessario invocare la funzione `display()` come mostrato di seguito:

```
$vei->display();
```

Questa istruzione va letta così: chiama la funzione `display()` dell'oggetto `$vei`. Lo script che segue aggiunge queste istruzioni al vostro script, che contiene la classe `veicolo`:

```
<?php

// Classi - Esempio 37-1
//---------------------

class veicolo {
  var $description;

  function set($d) {
    $this->description = $d;
  }

  function display() {
    echo($this->description);
  }
}

$vei = new veicolo;
$vei->set("La mia automobile nuova fiammante.");
$vei->display();

?>
```

L'output ottenuto da questo script è quello della Figura 37.3. Non molto interessante, ma è solo l'inizio!

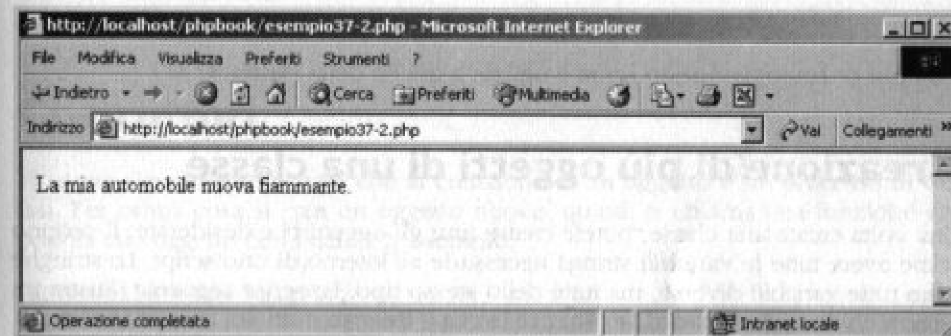


Figura 37.3
Output ottenuto da una classe.

Invocazione delle funzioni di classe dall'interno di una classe

Le funzioni di classe possono essere richiamate dall'interno di una classe. La sintassi necessaria è la seguente:

```
$this->nomeFunzione();
```

Questa funzionalità è molto utile poiché permette di invocare le funzioni automaticamente. Considerate l'esempio che segue, che è una variazione di quello precedente. Come potete vedere, la funzione `set()` contiene al suo interno una chiamata alla funzione `display()`: ciò consente di visualizzare automaticamente il valore di `description` quando si invoca la funzione `set()`, eliminando la necessità di invocare la funzione `display()` al di fuori dell'oggetto.

```
<?php

// Classi - Esempio 37-2
//---------------------

class veicolo {
```

```
    var $description;

    function set($d) {
        $this->description = $d;
        $this->display();
    }

    function display() {
        echo($this->description);
    }
}

$vei = new veicolo;
$vei->set("La mia automobile nuova fiammante.");

?>
```

L'output ottenuto da questo script è uguale a quello della Figura 37.3.

Creazione di più oggetti di una classe

Una volta creata una classe, potete creare tutti gli oggetti che desiderate. È proprio come avere tutte le variabili stringa necessarie all'interno di uno script. Le stringhe sono tutte variabili diverse, ma tutte dello stesso tipo. Lo script seguente illustra tre oggetti chiamati `$automobile`, `$bicicletta` e `$aereo`, tutti del tipo `veicolo`.

```
<?php

// Classi - Esempio 37-3
//----------------------

class veicolo {
   var $description;

   function set($d) {
      $this->description = $d;
      $this->display();
   }

   function display() {
   echo($this->description . "<br>");
   }
}

$automobile = new veicolo;
$automobile->set("Auto sportiva");
$bicicletta = new veicolo;
$bicicletta->set("Mountain bike");
$aereo = new veicolo;
$aereo->set("Biplano");

?>
```

L'output prodotto da questo script è quello della Figura 37.4.

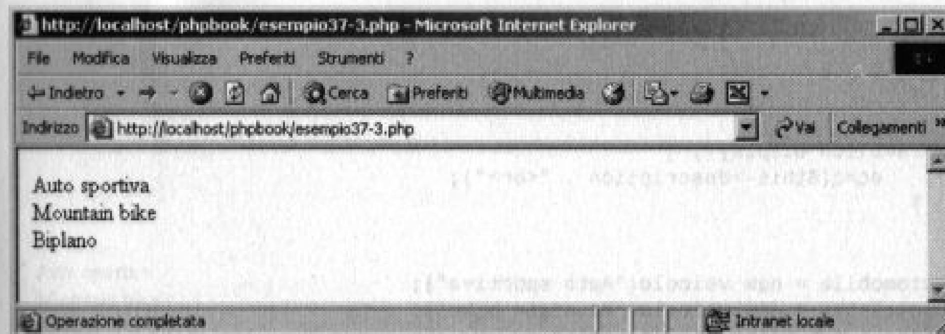


Figura 37.4
Più oggetti dello stesso tipo.

Costruttori

Vi sarete sicuramente accorti che la creazione di un oggetto è un processo in due fasi. Per prima cosa si crea un oggetto nuovo, quindi si chiama una funzione che popola tale oggetto con i dati. Per esempio:

```
$automobile = new veicolo;
$automobile->set("Auto sportiva");
```

Esiste un metodo per eseguire queste due operazione in un passaggio unico. A tal fine è necessario creare un costruttore, ossia una funzione che viene chiamata automaticamente quando si crea un nuovo oggetto. Una funzione diventa costruttore quando ha lo stesso nome della classe. Per esempio, una funzione costruttore per la classe `veicolo` avrebbe l'aspetto seguente:

```
function veicolo($d) {
   $this->description = $d;
   $this->display();
}
```

Il costruttore assegna il valore della variabile `$d` alla variabile `$description`, poi richiama la funzione `display()`. Con un costruttore a disposizione, potete associare la creazione dell'oggetto e la popolazione dei dati con la sintassi seguente:

```
$automobile= new veicolo("Auto sportiva");
```

Questa sintassi crea un nuovo oggetto di nome `$automobile` della classe `veicolo`, e passa la stringa "Auto sportiva" alla funzione costruttore. Lo script completo è quello che segue.

```
<?php

// Classi - Esempio 37-4
//----------------------

class veicolo {
   var $description;
```

```
  function veicolo($d) {
    $this->description = $d;
    $this->display();
  }

  function display() {
    echo($this->description . "<br>");
  }
}

$automobile = new veicolo("Auto sportiva");
$bicicletta = new veicolo("Mountain bike");
$aereo = new veicolo("Biplano");

?>
```

L'output ottenuto da questo script è uguale a quello della Figura 37.4.

Array di oggetti

Gli oggetti creati possono essere trattati come le variabili, pertanto è del tutto possibile creare un array di oggetti. Per esempio:

```
<?php

// Classi - Esempio 37-5
//

class veicolo {
  var $description;

  function veicolo($d) {
    $this->description = $d;
    $this->display();
  }

  function display() {
    echo($this->description . "<br>");
  }
}

$automobile = new veicolo("Auto sportiva");
$bicicletta = new veicolo("Mountain bike");
$aereo = new veicolo("Biplano");
$array = array($automobile,$bicicletta,$aereo);
foreach($array as $ilVeicolo)
  echo($ilVeicolo->display());

?>
```

Questo script mostra che i tre oggetti `veicolo` creati possono essere inseriti in un array (chiamato `$array`) e che a essi si può accedere con un ciclo `foreach`. L'output prodotto da questo script è quello della Figura 37.5.

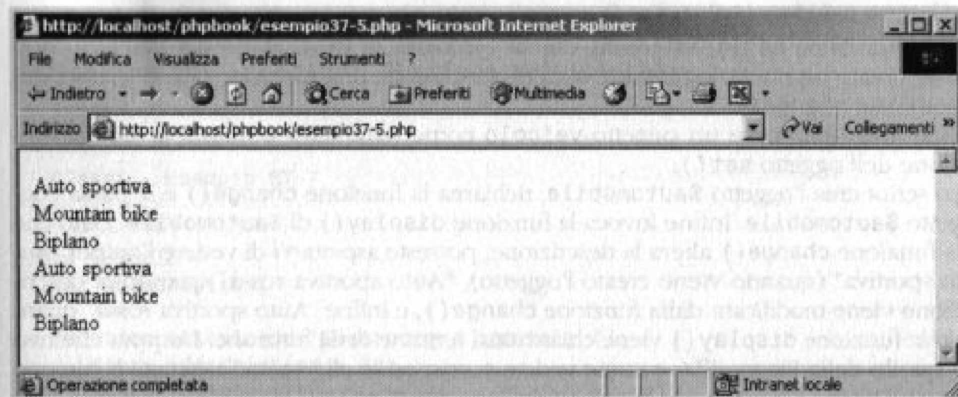


Figura 37.5
Utilizzo di oggetti in un array.

Funzioni e oggetti

Gli oggetti possono essere passati alle funzioni come avviene con le variabili. Lo script che segue mostra un problema cui occorre prestare attenzione.

```
<?php

// Classi - Esempio 37-6
//------------------------

class veicolo {
  var $description;

  function veicolo($d) {
    $this->description = $d;
    $this->display();
  }

  function set($d) {
    $this->description = $d;
    $this->display();
  }

  function display() {
    echo($this->description . "<br>");
  }
}

function change($vei){
  $vei->set("Auto sportiva rossa");
}

$automobile = new veicolo("Auto sportiva");
change($automobile);
$automobile->display();
```

Lo script contiene una classe `veicolo` modificata. Tale modifica riguarda l'aggiunta della funzione `set()`, che consente di cambiare la descrizione del veicolo dopo aver creato un nuovo oggetto. È stata inclusa anche una funzione non di classe, denominata `change()`.
Tale funzione riceve un oggetto `veicolo` come parametro e quindi invoca la funzione dell'oggetto `set()`.
Lo script crea l'oggetto `$automobile`, richiama la funzione `change()` e le passa l'oggetto `$automobile`. Infine invoca la funzione `display()` di `$automobile`. Dato che la funzione `change()` altera la descrizione, potreste aspettarvi di vedere l'output "Auto sportiva" (quando viene creato l'oggetto), "Auto sportiva rossa" quando la descrizione viene modificata dalla funzione `change()`, e infine "Auto sportiva rossa" quando la funzione `display()` viene chiamata al termine della funzione. L'output effettivo è quello della Figura 37.6 e come vedete corrisponde alle vostre aspettative, almeno fino all'ultima chiamata alla funzione dell'oggetto `display()`.

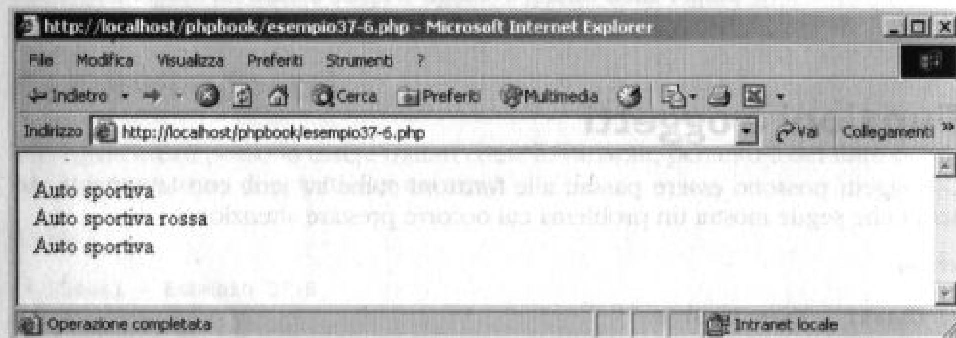


Figura 37.6
Output ottenuto dall'esempio di funzione.

L'oggetto sembra tornato alla sua descrizione originale. È accaduto, infatti, che alla funzione `change()` è stata inviata una copia dell'oggetto, ed è questa a essere stata modificata. Se volete che la funzione alteri l'originale, dovrete modificare la chiamata alla funzione nel modo seguente:

```
change(&$automobile);
```

La chiamata per riferimento è stata spiegata nel Capitolo 13.

Overload delle funzioni e argomenti predefiniti

Le funzioni di classe, al pari delle loro corrispondenti non di classe, possono includere valori come argomenti predefiniti. In tal modo una funzione può essere chiamata con o senza argomenti, se si sa con certezza che il valore predefinito mancante sarà impostato. La funzione costruttore dello script che segue mostra quanto appena affermato impostando la descrizione del veicolo a "Costruttore ignoto". Le classi non supportano l'overload delle funzioni. L'overload è il concetto secondo cui una classe può contenere più funzioni con lo stesso nome. Nello script che segue noterete che è stato necessario implementare due funzioni `set()` e `setu()`, dove una riceve un parametro e l'altra no. Nel caso della seconda funzione, il valore di `description` è impostato a `'Funzione set ignota'`.


```
<?php

// Classi - Esempio 37-7
//-------------------------

class veicolo {
   var $description;

   function veicolo($d='Costruttore ignoto') {
      $this->description = $d;
      $this->display();
   }

   function set($d) {
      $this->description = $d;
      $this->display();
   }

   function setu() {
      $this->description = 'Funzione set ignota';
          $this->display();
   }

   function display() {
      echo($this->description . "<br>");
   }
}

$automobile = new veicolo("Auto sportiva");
$automobile2 = new veicolo();
$automobile2->setu();

?>
```

L'output prodotto da questo script è quello della Figura 37.7.

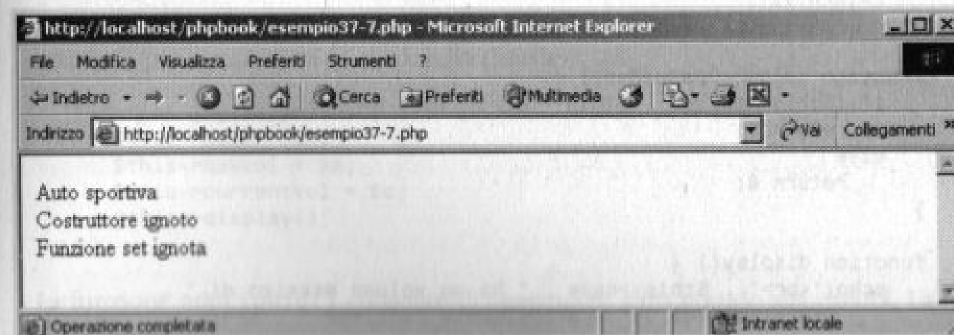


Figura 37.7
Output dell'overload di funzione.

Creazione di una classe più complessa

Finora avete visto esempi di classi molto semplici per capire le funzionalità a oggetti supportate da PHP. L'esempio seguente illustra una classe leggermente più complessa e qualche iterazione tra gli oggetti. Verrà creata una classe di nome `jug` che permette di modellare la funzionalità di una caraffa. Ogni oggetto `jug` può contenere una certa quantità di liquido.

Ma l'esempio non finisce qui. Saranno create due caraffe e ognuna di esse potrà versare una parte o tutto il proprio contenuto nell'altra.

```
<?php

// Classi - Esempio 37-8
//----------------------

class jug {
  var $maxVol;
  var $currentVol;
  var $name;

  function jug($n,$m,$c) {
    $this->name = $n;
    $this->maxVol = $m;
    $this->currentVol = $c;
    $this->display();
  }

  function addLiquid($v){
    $this->currentVol = $this->currentVol + $v;
  }

  function name(){
    return $this->name;
  }

  function isThereLiquid($vol) {
    if($this->currentVol >= $vol)
      return 1;
  else
    return 0;
  }

  function isThereSpace($vol) {
    if(($this->maxVol - $this->currentVol) >= $vol)
      return 1;
    else
      return 0;
  }

  function display() {
    echo("<br>" . $this->name . " ha un volume massimo di " .
      $this->maxVol . " e attualmente contiene " . $this->currentVol);
  }

  function pourInTo($otherJug,$vol) {
    echo("<br>Versamento di $vol unità da " . $this->name() . " in " .
      $otherJug->name());
    if($this->isThereLiquid($vol)) {
      if($otherJug->isThereSpace($vol)) {
        $otherJug->addLiquid($vol);
        $this->currentVol = $this->currentVol - $vol;
        $this->display();
        $otherJug->display();
      }
      else {
        echo("<br>Tentativo fallito - Spazio insufficiente.");
        return 0;
      }
    }
    else {
      echo("<br>Tentativo fallito - Liquido insufficiente.");
      return 0;
    }
    return 1;
  }
}

$myJug = new jug("Caraffa 1",100,50);
$anotherJug = new jug("Caraffa 2",50,25);
$myJug->pourInTo(&$anotherJug,10);
$anotherJug->pourInTo(&$myJug,34);

?>
```

Lo script inizia creando una classe di nome `jug`. La classe è costituita da tre variabili che servono per memorizzare il volume massimo di liquido che la caraffa può contenere, la quantità corrente di liquido nella caraffa e il nome della caraffa (che serve solo a identificarla):

```
<?php

class jug {
  var $maxVol;
  var $currentVol;
  var $name;
```

Il costruttore crea l'oggetto con le tre variabili che riceve e invoca la funzione `display()` per visualizzare lo stato della caraffa:

```
  function jug($n,$m,$c) {
    $this->name = $n;
    $this->maxVol = $m;
    $this->currentVol = $c;
    $this->display();
  }
```

La funzione `addLiquid()` consente di aggiungere altro liquido alla caraffa. La funzione `name()` restituisce il nome della caraffa:

```
  function addLiquid($v){
    $this->currentVol = $this->currentVol + $v;
```

```
}

function name(){
   return $this->name;
}
```

La funzione `isThereLiquid()` verifica se il liquido contenuto nella caraffa è uguale o supera il valore del volume passato alla funzione in `$vol`. Se c'è liquido a sufficienza, viene restituito il valore 1, altrimenti 0:

```
function isThereLiquid($vol) {
   if($this->currentVol >= $vol)
      return 1;
   else
      return 0;
}
```

La funzione `isThereSpace()` verifica se nella caraffa c'è spazio sufficiente per il volume di liquido passato alla funzione in `$vol`. In caso affermativo, viene restituito il valore 1, altrimenti 0:

```
function isThereSpace($vol) {
   if(($this->maxVol - $this->currentVol) >= $vol)
      return 1;
   else
      return 0;
}
```

La funzione `display()` visualizza la quantità di liquido che la caraffa è in grado di contenere e la quantità attualmente contenuta:

```
function display() {
   echo("<br>" . $this->name . " ha un volume massimo di " .
        $this->maxVol . " e attualmente contiene " . $this->currentVol);
}
```

La funzione `pourInTo()` è la più complessa tra quelle incontrate finora, e pertanto richiede di essere suddivisa in sezioni più piccole. Per prima cosa, la funzione accetta due argomenti, `$otherJug` e `$vol`.
Uno rappresenta l'oggetto `jug` in cui sarà riversato il liquido e l'altro il numero di unità da versare:

```
function pourInTo($otherJug,$vol) {
```

Successivamente, la funzione visualizza il numero di unità che saranno versate da questa caraffa nell'altra. Con una chiamata alla funzione `$otherJug->name()` si ottiene il nome dell'altra caraffa:

```
echo("<br>Versamento di $vol unità da " . $this->name() . " in " .
   $otherJug->name());
```

Quindi, una chiamata alla funzione `isThereLiquid()` accerta se la caraffa contiene liquido a sufficienza da versare nell'altra caraffa:

```
if($this->isThereLiquid($vol)) {
```


Se il liquido è sufficiente, la funzione `isThereSpace()` di `otherJug` viene invocata per verificare se l'altra caraffa può accettare il liquido. In caso affermativo, la funzione `addLiquid()` di `otherJug` è richiamata per aggiungere il liquido e il volume corrente di liquido in questa caraffa diminuisce. Infine, compaiono i contenuti delle due caraffe:

```
      if($otherJug->isThereSpace($vol)) {
            $otherJug->addLiquid($vol);
            $this->currentVol = $this->currentVol - $vol;
            $this->display();
            $otherJug->display();
      }
```

Se non c'è spazio sufficiente, viene generato un messaggio d'errore:

```
      else {
         echo("<br>Tentativo fallito - Spazio insufficiente.");
         return 0;
      }
```

Se il liquido non basta, viene generato un messaggio d'errore:

```
   else {
      echo("<br>Tentativo fallito - Liquido insufficiente.");
      return 0;
   }
}
```

Sono state create due caraffe: la prima, Caraffa 1, ha un volume massimo di 100 unità ed è piena a metà. Caraffa 2 ha una capacità massima di 50 unità ed è anch'essa piena a metà:

```
$myJug = new jug("Caraffa 1",100,50);
$anotherJug = new jug("Caraffa 2",50,25);
```

Viene invocata la funzione `pourInTo()` dell'oggetto `myJug` e le vengono passati i valori `$anotherJug` e `10`. In questo modo, `myJug` cercherà di versare 10 unità in `anotherJug`:

```
$myJug->pourInTo(&$anotherJug,10);
```

Infine, viene invocata la funzione `pourInTo()` dell'oggetto `anotherJug` e le vengono passati i valori `$myJug` e `34`. In questo modo, `anotherJug` cercherà di versare 34 unità in `myJug`:

```
$anotherJug->pourInTo(&$myJug,34);

?>
```

L'output prodotto da questo script è quello della Figura 37.8.
La Figura 37.8 illustra la creazione di entrambe le caraffe e le interazioni tra esse. Dopo che da Caraffa 1 è stato versato un po' di liquido in Caraffa 2, e viceversa, il volume di liquido nelle due caraffe è cambiato da 50 e 25 a 1 e 74.

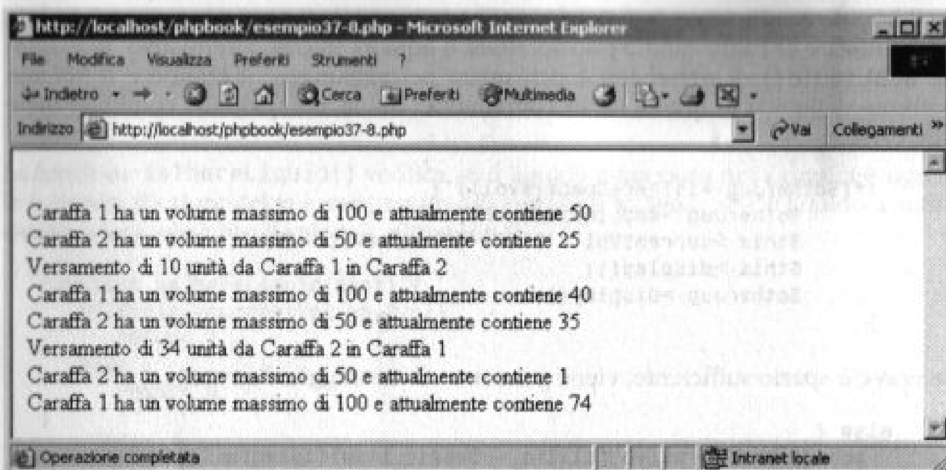


Figura 37.8
Output dell'oggetto jug.

Oggetti all'interno delle classi

Si è già visto come gli oggetti possano invocare le proprie funzioni e quelle degli altri oggetti.
Non sapete ancora, però, che si possono creare oggetti che contengono altri oggetti. In qualche caso questi oggetti sono indicati con il nome di oggetti compositi. Nell'esempio sarà creata una classe denominata **serratura** e una classe composita di nome **porta** che conterrà una serratura. Potete creare oggetti di tipo **serratura** separati (e sarebbe opportuno che lo faceste), ma quando create una porta non potete fare a meno di creare anche una serratura.
Iniziate con la classe **serratura** (che contiene un'unica variabile per memorizzare se la serratura è chiusa oppure no), un costruttore per configurare la serratura e una funzione **display()** per visualizzare lo stato della serratura:

```
<?php

// Classi - Esempio 37-9
//----------------------

class serratura {
  var $lockUnlock;

  function serratura($lu) {
    $this->lockUnlock = $lu;
  }

  function display() {
    echo("<br>La serratura è " . $this->lockUnlock);
  }
}
```


La classe **porta** è un po' più complessa (ma non molto). Contiene due variabili: la prima, **$openClosed**, serve per memorizzare se la porta è chiusa o aperta; la seconda, **$serratura**, memorizza l'oggetto **serratura** che andrete a creare:

```
class porta {
  var $openClosed;
  var $serratura;
```

Il costruttore riceve due parametri. Il primo, **$oc**, serve per impostare il valore della variabile **$openClosed**. Il secondo, **$lu**, viene passato al costruttore della classe **serratura** quando create l'oggetto **serratura**:

```
function porta($oc,$lu){
  $this->openClosed = $oc;
  $this->serratura = new serratura($lu);
  $this->display();
}
```

Quando alla variabile **$serratura**, che si trova all'interno della classe **porta**, viene assegnato un valore, si tratta effettivamente di un'istanza di un nuovo oggetto **serratura**. La funzione **display()** visualizza il valore della variabile **$openClosed** all'interno della classe **porta**. Quindi, invoca la funzione **display** dell'oggetto **serratura** che appartiene alla classe **serratura**. Questa chiamata richiede due operatori **->**:

```
  function display(){
    echo("<br>La porta è " . $this->openClosed);
    $this->serratura->display();
  }
}

$miaPorta = new porta("chiusa","aperta");

?>
```

L'output ottenuto da questo script è quello della Figura 37.9.

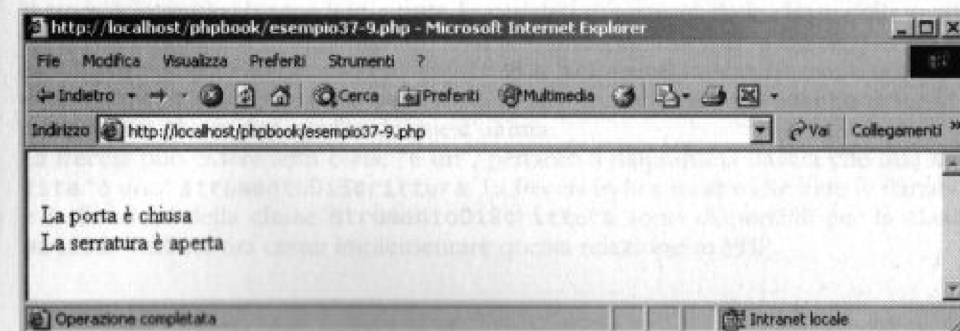


Figura 37.9
Output di oggetti all'interno di altri oggetti.

Come vedete, **porta** è un oggetto composito poiché al momento della sua creazione è stato generato anche un oggetto **serratura**.

Riepilogo

Questo capitolo ha introdotto il concetto di orientamento agli oggetti, spiegando la differenza tra classi e oggetti e mostrando come utilizzarli in PHP.
Avete imparato a creare qualche semplice classe e avete visto come vengono costruite le variabili e le funzioni di classe. Inoltre, avete appreso il concetto di incapsulamento e visto come le interazioni tra variabili debbano avvenire attraverso le funzioni di classe. Infine, avete appreso il concetto di oggetto composito. Tuttavia, l'argomento della programmazione a oggetti non è esaurito. L'orientamento agli oggetti fornisce infatti una funzionalità chiave per il riutilizzo del software. Questa funzionalità è nota con il nome di ereditarietà e sarà l'argomento del prossimo capitolo.

Capitolo 38

Ereditarietà delle classi

Introduzione

Nel capitolo precedente sono stati introdotti i concetti di orientamento agli oggetti, classi e oggetti. Si è visto come PHP possa essere impiegato per creare classi e come sia possibile usare le istanze di queste classi per creare oggetti. All'inizio del capitolo precedente si è visto come uno dei vantaggi dell'orientamento agli oggetti sia il suo supporto al riutilizzo del codice. In questo senso, la caratteristica principale dell'orientamento agli oggetti è l'ereditarietà delle classi. Questo capitolo introdurrà l'ereditarietà delle classi, illustrando in che modo e perché si riveli utile. Verrà introdotto anche il concetto di polimorfismo, che è correlato all'ereditarietà.

Che cos'è l'ereditarietà delle classi

Uno dei punti di forza dell'orientamento agli oggetti è la capacità di ereditare proprietà da altre classi già esistenti. Ciò comporta un risparmio di tempo per il programmatore, riduce la complessità della soluzione e aiuta a migliorare la qualità riducendo la duplicazione del codice, e pertanto l'introduzione di errori. Il programmatore è in grado di accedere a tutte le variabili e funzioni della classe dalla quale vengono ereditate e di aggiungere variabili e funzioni alla nuova classe. Un esempio di ereditarietà è illustrato nella Figura 38.1. Nell'esempio sono presenti due classi, `strumentoDiScrittura` e `matita`; la freccia da `matita` a `strumentoDiScrittura` indica che matita eredita da quest'ultima.
La freccia può essere letta come "è un", pertanto il diagramma illustra che una `matita` "è uno" `strumentoDiScrittura`. La freccia indica inoltre che tutte le variabili e le funzioni della classe `strumentoDiScrittura` sono disponibili per la classe `matita`. Vedrete ora come implementare questa relazione in PHP.

Parola chiave extends

La parola chiave `extends` viene utilizzata per indicare che una classe eredita da un'altra. La sintassi è la seguente.

```
class nomeClasse extends nomeAltraClasse
```

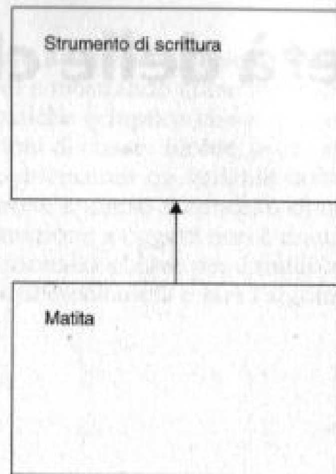


Figura 38.1
Ereditarietà.

Per implementare le due classi della Figura 38.1 occorre scrivere quanto segue:

```
class strumentoDiScrittura {
}

class matita extends strumentoDiScrittura {
}
```

Ovviamente questi sono solo i modelli delle classi senza alcuna variabile o funzione. Occorre fornire un contenuto alle classi e quindi, per esempio, la classe `strumentoDiScrittura` potrebbe essere implementata nel modo seguente:

```
class strumentoDiScrittura {
   var $textColour;

   function strumentoDiScrittura($t) {
      $this->textColour = $t;
   }

   function display() {
      echo("il colore è " . $this->textColour . "<br>");
   }
}
```

La classe `strumentoDiScrittura` contiene una singola variabile `$textColour`, che verrà utilizzata per memorizzare il colore del testo prodotto dallo strumento di scrittura. La classe contiene inoltre un costruttore per creare il colore del testo dell'oggetto e una funzione `display()` per visualizzare il valore del colore del testo.

Ora siete pronti per implementare la classe `matita`. Inizialmente la manterrete il più semplice possibile:

```
class matita extends strumentoDiScrittura {

   function matita ($t) {
      $this->strumentoDiScrittura($t);
   }
}
```

La classe `matita` è molto semplice: non contiene variabili proprie e la sua unica funzione è un costruttore. Normalmente, le classi che ereditano da altre classi hanno bisogno di un costruttore poiché, nel caso in cui venga creata un'istanza della classe, è necessario poter costruire il valore delle variabili nella classe ereditata. In questo esempio il costruttore della classe richiama semplicemente il costruttore della classe `strumentoDiScrittura`, passandole il valore del colore della mina della matita. Osservate che il costruttore della classe `strumentoDiScrittura` viene chiamato come se fosse un membro della classe `matita` con la sintassi seguente:

```
$this->strumentoDiScrittura($t);
```

Questo avviene perché tutte le funzioni, compreso il costruttore della classe ereditata, fanno parte della classe `matita`.
Osservando meglio la classe `matita` potete notare che la classe di per sé costituisce un'ereditarietà piuttosto inutile, in quanto la classe `matita` non fa niente di diverso dalla classe `strumentoDiScrittura`. È quindi necessario estendere la funzionalità della classe `matita` così che operi diversamente dalla classe `strumentoDiScrittura` e pertanto diventi utile.

Aggiunta di nuove funzioni e variabili

Nel paragrafo precedente è stata creata una classe `matita` che eredita dalla classe `strumentoDiScrittura`. Sfortunatamente la classe `matita` non includeva alcuna variabile o funzione propria che la rendesse diversa dalla classe ereditata. È quindi il momento di intervenire:

```
class matita extends strumentoDiScrittura {
   var $sharp;

   function matita ($t,$s) {
      $this->sharp = $s;
      $this->strumentoDiScrittura($t);
      $this->display2();
   }
   function display2() {
      echo("<br>La matita è " . $this->sharp . " e ");
      $this->display();
   }
}
```

Ora la classe `matita` ha una variabile `$sharp`, che verrà utilizzata per memorizzare se la matita ha la punta o è spuntata. Il costruttore è stato modificato per creare la

variabile `$sharp` e per richiamare la classe `strumentoDiScrittura`, dove verrà costruito il colore del testo. Infine, è stata inclusa una funzione `display2()` per visualizzare il valore della variabile `$sharp` e per invocare la funzione `display()` di `strumentoDiScrittura`.
Ora è possibile creare oggetti di tipo `strumentoDiScrittura` e `matita`:

```
$thing = new strumentoDiScrittura("blu");
$thing->display();
$myGreyPencil = new matita("grigio","appuntita");
```

Il listato completo è il seguente:

```
<?php

// Ereditarietà delle classi - Esempio 38-1
//-----------------------------------

class strumentoDiScrittura {
   var $textColour;

   function strumentoDiScrittura($t) {
      $this->textColour = $t;
   }

   function display() {
      echo("il colore è " . $this->textColour . "<br>");
   }
}

class matita extends strumentoDiScrittura {
   var $sharp;

   function matita($t,$s) {
      $this->sharp = $s;
      $this->strumentoDiScrittura($t);
      $this->display2();
   }

   function display2() {
      echo("<br>La matita è " . $this->sharp . " e ");
   $this->display();
   }
}

$thing = new strumentoDiScrittura("blu");
$thing->display();
$myGreyPencil = new matita("grigio","appuntita");

?>
```

La combinazione di queste istruzioni in uno script produrrà l'output della Figura 38.2. La Figura 38.2 mostra che l'output è diverso per i due oggetti. Tramite l'ereditarietà siete riusciti a creare una nuova classe `matita`, che riutilizza parte del codice della classe `strumentoDiScrittura`. Facendo questo non avete alterato in alcun modo la classe `strumentoDiScrittura` e avete solo stabilito quali sono le differenze tra le due classi.

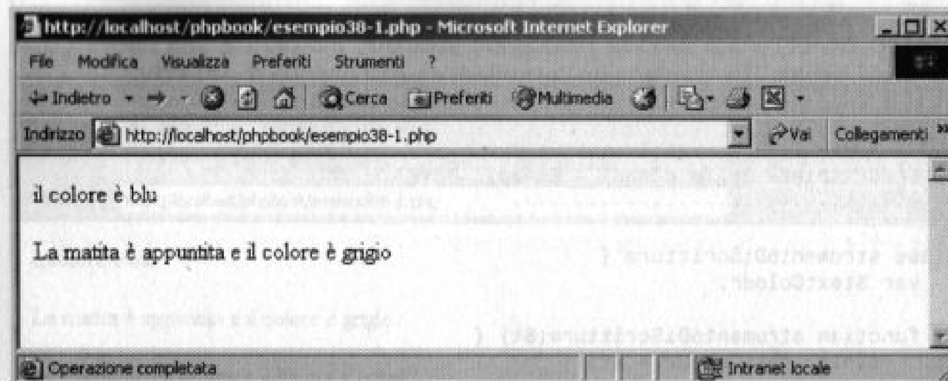


Figura 38.2
Output ottenuto con l'ereditarietà.

Ulteriore estensione dell'ereditarietà

Una volta creata una struttura di ereditarietà, potete continuare a estenderla a vostro piacimento. La Figura 38.3 mostra una classe aggiuntiva di nome `penna`, che eredita anch'essa da `strumentoDiScrittura`. Volendo potreste aggiungere una classe che eredita dalla classe `matita`, per esempio una classe chiamata `Supermatita sempre appuntita`, che avrebbe quindi ereditato tutti gli attributi della classe `matita` e quelli di `strumentoDiScrittura`; per il momento, però, la classe `penna` è sufficiente.

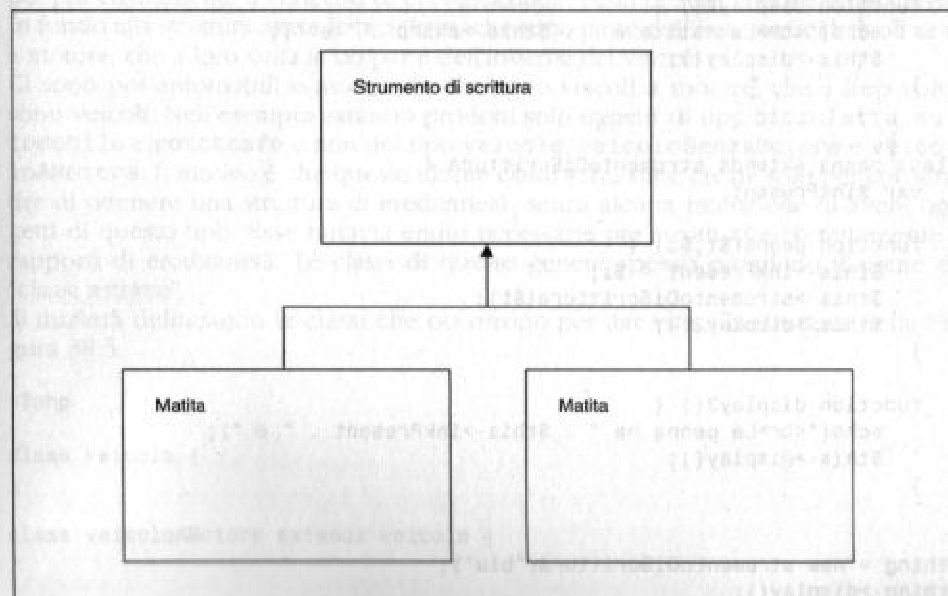


Figura 38.3
Estensione dell'ereditarietà.

Lo script che segue sviluppa l'implementazione della Figura 38.3. Le classi `strumentoDiScrittura` e `matita` rimangono invariate, ma ora c'è una nuova classe chiamata `penna`, che contiene un costruttore e una funzione `display2()`:

```
<?php

// Ereditarietà delle classi - Esempio 38-2
//-----------------------------------

class strumentoDiScrittura {
   var $textColour;

   function strumentoDiScrittura($t) {
      $this->textColour = $t;
   }

   function display() {
      echo("il colore è " . $this->textColour . "<br>");
   }
}

class matita extends strumentoDiScrittura {
   var $sharp;

   function matita($t,$s) {
      $this->sharp = $s;
      $this->strumentoDiScrittura($t);
      $this->display2();
   }

   function display2() {
      echo("<br>La matita è " . $this->sharp . " e ");
      $this->display();
   }
}

class penna extends strumentoDiScrittura {
   var $inkPresent;

   function penna($t,$i) {
      $this->inkPresent = $i;
      $this->strumentoDiScrittura($t);
      $this->display2();
   }

   function display3() {
      echo("<br>La penna ha " . $this->inkPresent . " e ");
      $this->display();
   }
}

$thing = new strumentoDiScrittura("blu");
$thing->display();
$myGreyPencil = new matita("grigio","appuntita");
$myRedPen = new penna("rosso","inchiostro")
?>
```

Vengono costruiti e visualizzati tre oggetti e il loro output è quello della Figura 38.4.

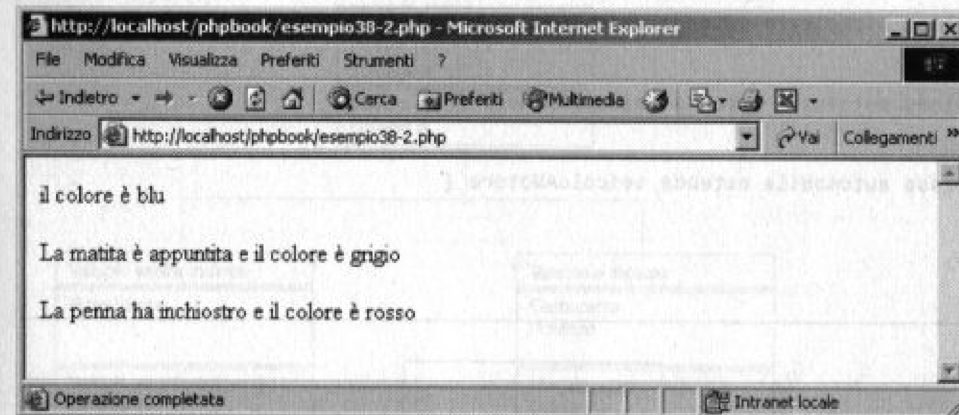


Figura 38.4
Estensione dell'ereditarietà.

Un esempio più complesso di ereditarietà

Ora si esaminerà a un esempio costituito da un grande numero di classi. Considerate la struttura di ereditarietà della Figura 38.5, costituita da sei classi.

La struttura di ereditarietà della Figura 38.5 è più complessa di prima e illustra un po' più chiaramente il concetto di classificazione, ossia di suddivisione delle classi. In fondo alla struttura avete le biciclette, che sono parte dell'insieme dei veicoli non a motore, che a loro volta sono parte dell'insieme dei veicoli.

Ci sono poi automobili e motoscafi, che sono veicoli a motore, che a loro volta sono veicoli. Nell'esempio saranno prodotti solo oggetti di tipo `bicicletta`, `automobile` e `motoscafo` e non del tipo `veicolo`, `veicoloSenzaMotore` e `veicoloAMotore`. Il motivo è che queste ultime classi sono state create solo per consentire di ottenere una struttura di ereditarietà, senza alcuna intenzione di avere oggetti di questo tipo. Esse tuttavia erano necessarie per strutturare correttamente i rapporti di ereditarietà. Le classi di questo genere spesso prendono il nome di "classi astratte".

Si inizierà delineando le classi che occorrono per dar vita alla struttura della Figura 38.5.

```
<?php

class veicolo {
}

class veicoloAMotore extends veicolo {
}
```

```
class veicoloSenzaMotore extends veicolo {
}

class bicicletta extends veicoloSenzaMotore {
}

class motoscafo extends veicoloAMotore {
}

class automobile extends veicoloAMotore {
}

?>
```

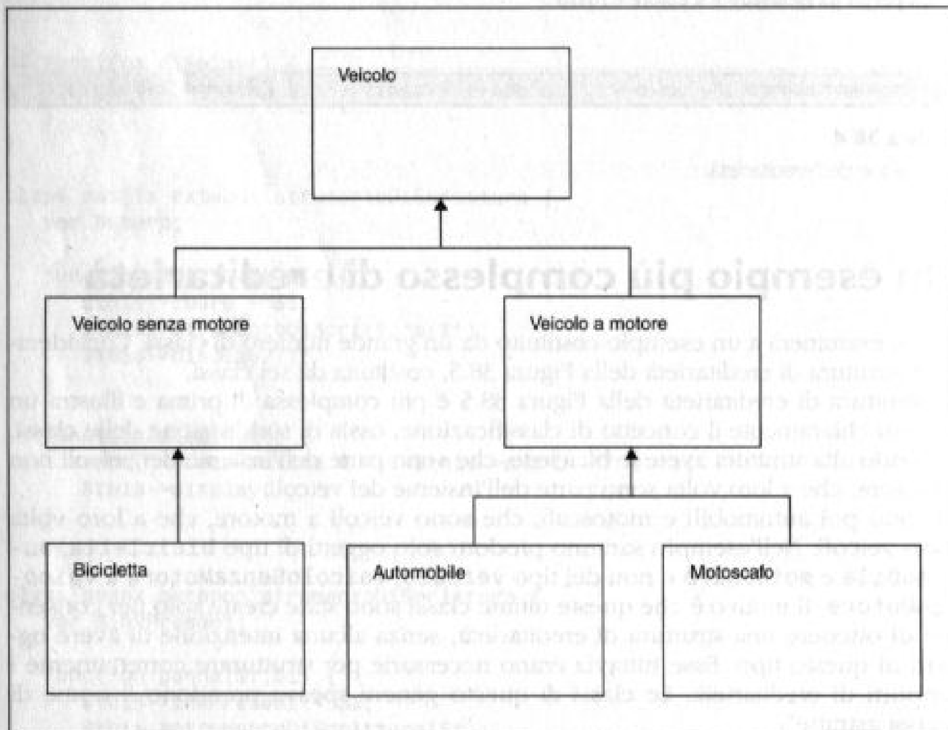


Figura 38.5
Struttura di ereditarietà dei veicoli.

Questa parte di codice fornisce i modelli delle classi che andrete a creare. A questo punto è necessario determinare le funzioni e le variabili che ciascuna classe conterrà. La Figura 38.6 migliora la struttura della Figura 38.5 e fornisce alcuni dettagli sulle variabili e le funzioni che ciascuna delle classi conterrà.

Per semplicità, ogni classe contiene un costruttore e una funzione `display()` che visualizza i valori delle sue variabili. Ogni classe contiene da una a tre variabili. La classe `veicolo` contiene tre variabili per memorizzare una descrizione del veicolo, il colore e il nome del file di immagine da visualizzare per rappresentare il veicolo.

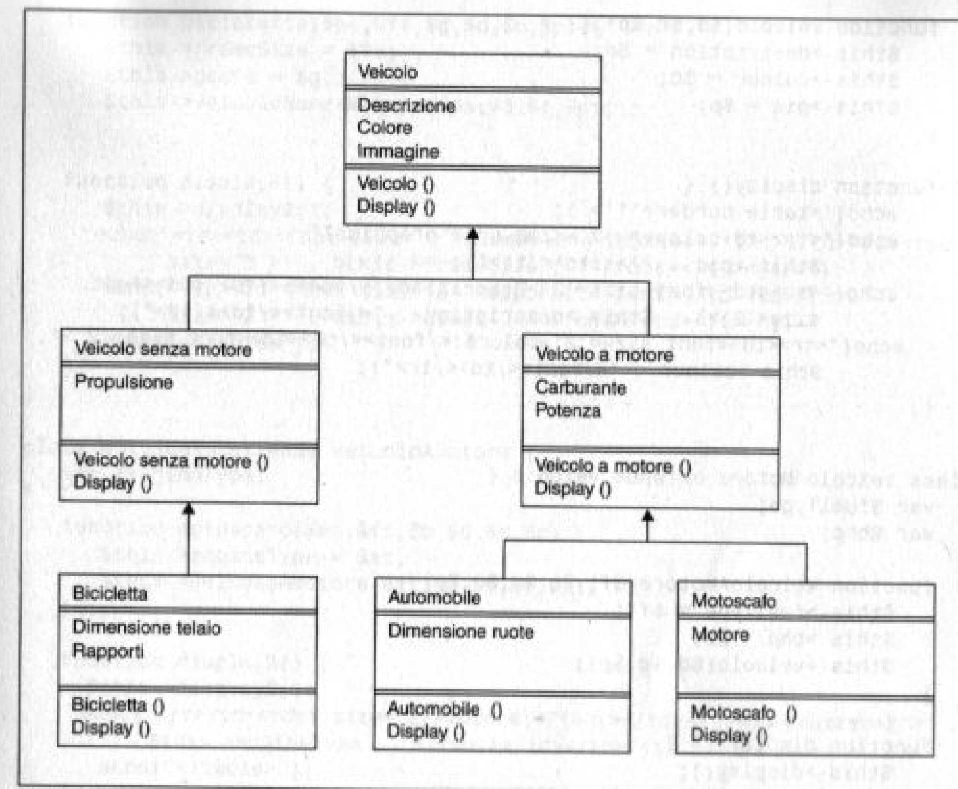


Figura 38.6
Struttura di ereditarietà dei veicoli con metodi delle classi e dati.

La classe `veicoloSenzaMotore` ha una variabile `powerSource`, che memorizza la fonte di propulsione del veicolo: vento, umana e così via.

La classe `veicoloAMotore` contiene due variabili per memorizzare il tipo di carburante (benzina, gasolio) e il valore BHP (potenza effettiva) del veicolo. La classe `bicicletta` contiene due variabili che memorizzano la dimensione del telaio e il numero di rapporti.

La classe `automobile` contiene una variabile che memorizza la dimensione delle ruote del veicolo. Infine la classe `motoscafo` ha una variabile che memorizza se il motore è entrobordo o fuoribordo.

Lo script seguente implementa lo schema della Figura 38.6.

```
<?php

// Ereditarietà delle classi - Esempio 38-3
//-----------------------------------

class veicolo {
   var $description;
   var $colour;
   var $pic;
```

```
  function veicolo($d,$c,$p) {
     $this->description = $d;
     $this->colour = $c;
     $this->pic = $p;
  }

  function display() {
     echo("<table border='1'>");
     echo("<tr><td colspan='2'><img src='graphics/" .
          $this->pic . "'></td></tr>");
     echo("<tr><td><font size='2'>Descrizione:</font></td> <td><font
          size='2'>" . $this->description . "</font></td></tr>");
    echo("<tr><td><font size='2'>Colore:</font></td><td><font size='2'>".
          $this->colour . "</font></td></tr>");
  }
}

class veicoloAMotore extends veicolo {
  var $fuelType;
  var $bhp;

  function veicoloAMotore($ft,$b,$d,$c,$p) {
     $this->fuelType = $ft;
     $this->bhp = $b;
     $this->veicolo($d,$c,$p);
  }

  function display2() {
     $this->display();
     echo("<tr><td><font size='2'>Tipo di
          carburante:</font></td><td><font size='2'>" . $this->fuelType .
          "</font></td></tr>");
     echo("<tr><td><font size='2'>Potenza:</font></td>
          <td><font size='2'>" . $this->bhp . "</font></td></tr>");
  }
}
class veicoloSenzaMotore extends veicolo {
  var $powerSource;

  function veicoloSenzaMotore($ps,$d,$c,$p) {
     $this->powerSource = $ps;
     $this->veicolo($d,$c,$p);
  }

  function display2() {
     $this->display();
     echo("<tr><td><font size='2'>Propulsione:</font></td><td><font
          size='2'>" . $this->powerSource . "</font></td></tr>");
  }
}

class bicicletta extends veicoloSenzaMotore {
  var $frameSize;
  var $gears;

  function bicicletta($ps,$fs,$g,$d,$c,$p) {
     $this->frameSize = $fs;
     $this->gears = $g;
     $this->veicoloSenzaMotore($ps,$d,$c,$p);
  }

  function display3() {
     $this->display2();
    echo("<tr><td><font size='2'>Dimensione telaio:</font></td><td><font
          size='2'>" . $this->frameSize . "</font></td></tr>");
     echo("<tr><td><font size='2'>Rapporti:</font></td><td><font
          size='2'>" . $this->gears . "</font></td></tr>");
     echo("</table>");
  }
}

class motoscafo extends veicoloAMotore {
  var $engineType;

  function motoscafo($et,$ft,$b,$d,$c,$p) {
     $this->engineType = $et;
     $this->veicoloAMotore($ft,$b,$d,$c,$p);
  }

  function display3() {
    $this->display2();
   echo("<tr><td><font size='2'>Motore:</font></td><td><font size='2'>" .
          $this->engineType . "</font></td></tr>");
     echo("</table>");
  }
}

class automobile extends veicoloAMotore {
  var $wheelSize;

  function automobile($ws,$ft,$b,$d,$c,$p) {
     $this->wheelSize = $ws;
     $this->veicoloAMotore($ft,$b,$d,$c,$p);
  }

  function display3() {
     $this->display2();
     echo("<tr><td><font size='2'>Dimensione ruote:</font></td><td><font
          size='2'>" . $this->wheelSize . "</font></td></tr>");
     echo("</table>");
  }
}

$veh = new
motoscafo("Fuoribordo","Diesel",60,"Barca","Giallo","speedboat.jpg");
$veh2 = new bicicletta("Umana",18,21,"Bicicletta","Verde","bicycle.jpg");
$veh3 = new automobile(17,"Benzina",147,"Automobile","Rosso","car.jpg");
$vehicles = array ($veh,$veh2,$veh3);
foreach ($vehicles as $vehicle){
```

```
    $vehicle->display3();
}

?>
```

Questo script dichiara tre oggetti, un motoscafo, una bicicletta e un'automobile. Questi vengono copiati in un array e viene utilizzato un ciclo `foreach` per visualizzare ciascun oggetto invocando la funzione `display3()`. Lo script utilizza tre immagini, che saranno visualizzate assieme ai dati del veicolo per rendere più interessante l'output dello script. La Tabella 38.1 elenca queste immagini e i loro nomi di file.

Tabella 38.1 Immagini dei veicoli

	Bicycle.jpg
	Speedboat.jpg
	Car.jpg

L'output prodotto da questo script è quello della Figura 38.7. Osservate che il ciclo `foreach` accede a ciascuno degli oggetti dell'array e li visualizza richiamando una singola funzione.
Tutto funziona bene, ma c'è un problema che dovete risolvere se volete poter sfruttare tutta la potenza dell'ereditarietà. Si tratta del problema che viene analizzato nel prossimo paragrafo, dedicato al polimorfismo.

Polimorfismo

Avrete notato che ciascuna classe in una ramificazione della struttura di ereditarietà ha tre funzioni `display()`, ciascuna delle quali ha un nome diverso a ogni livello: `display()`, `display2()` e `display3()`. Ciò consente di chiamare la funzione nella prima classe a risalire nella struttura.

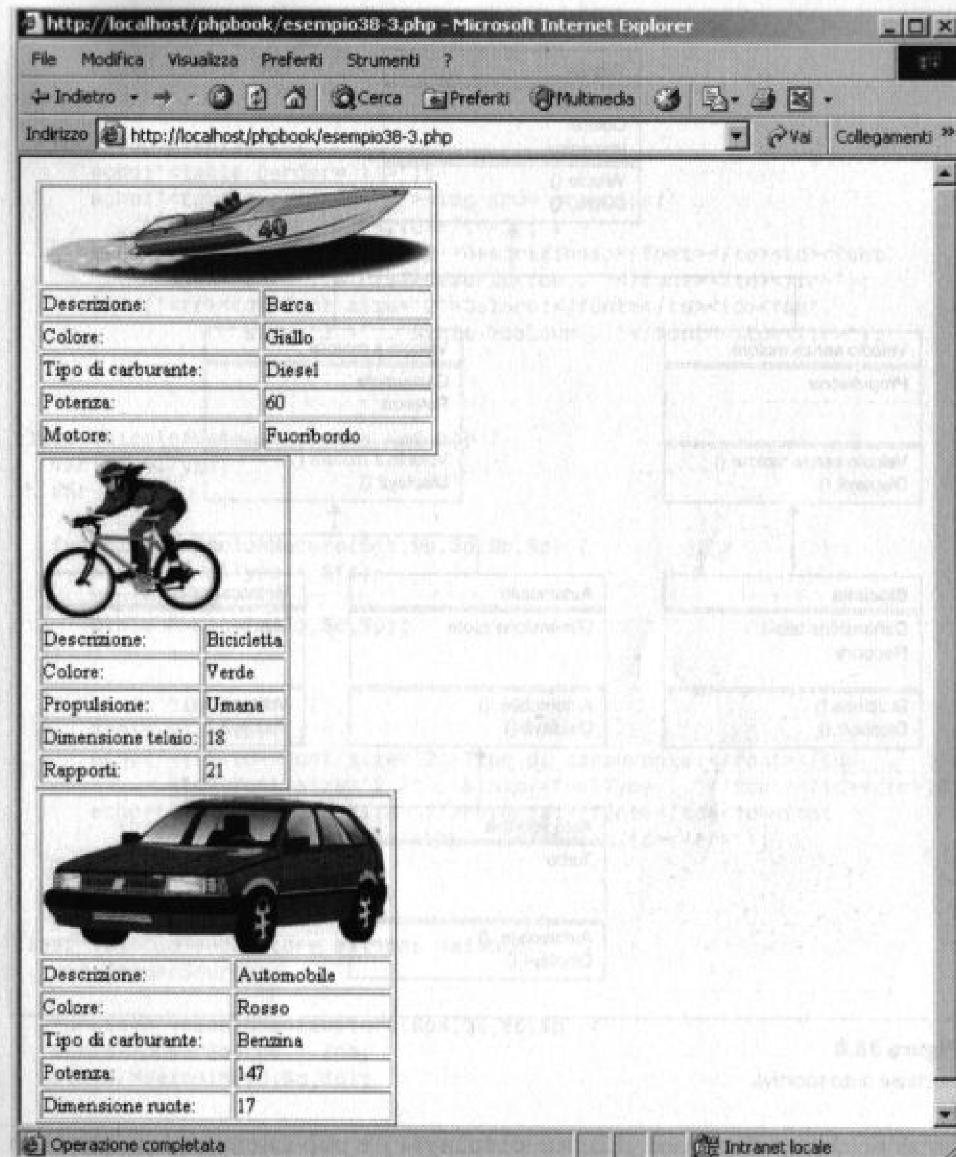


Figura 38.7
Output dell'ereditarietà dei veicoli.

Questo non determina alcun problema nel progetto corrente, ma cosa accadrebbe se voleste aggiungere una nuova classe, come mostrato nella Figura 38.8?
La Figura 38.8 segnala che ora avete una nuova classe chiamata `autoSportiva`. La classe ha una variabile `turbo` che memorizza la pressione in bar del turbocompressore (presupponendo che tutte le auto sportive ne abbiano uno). La classe ha una

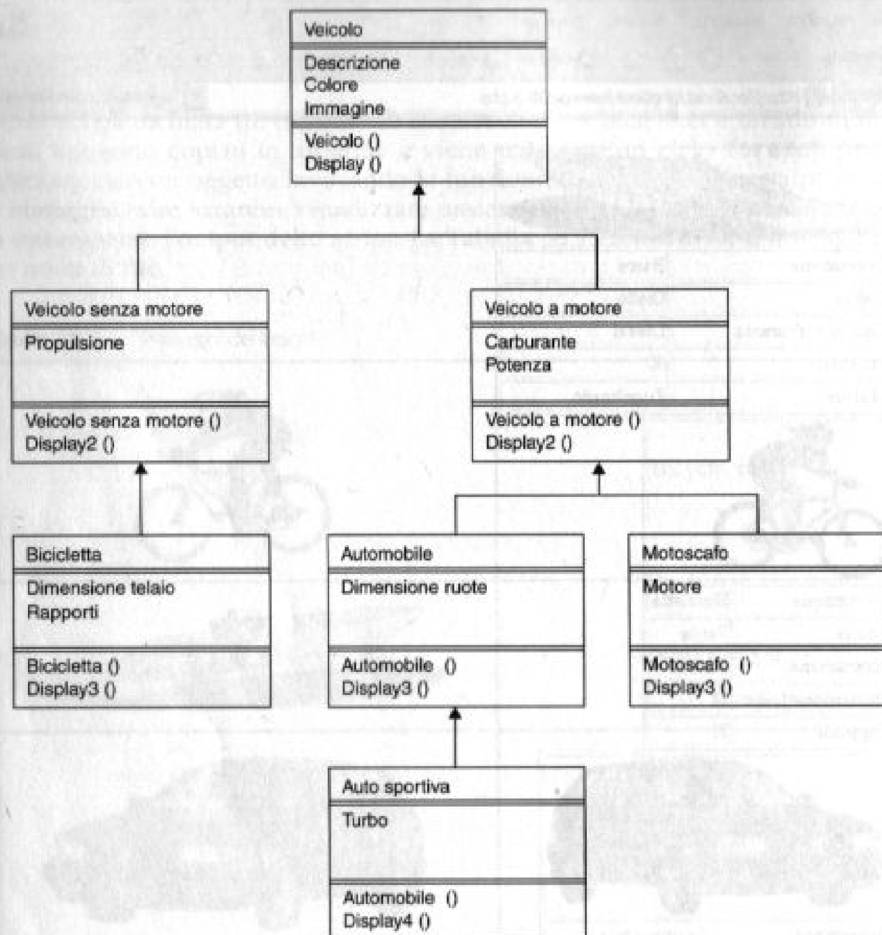


Figura 38.8
La classe auto sportiva.

funzione di visualizzazione chiamata **display4()** e può essere implementata con relativa facilità.

```
<?php

// Ereditarietà delle classi - Esempio 38-4
//-----------------------------------

class veicolo {
  var $description;
  var $colour;
  var $pic;

  function veicolo($d,$c,$p) {
    $this->description = $d;
    $this->colour = $c;
    $this->pic = $p;
  }

  function display() {
    echo("<table border='1'>");
    echo("<tr><td colspan='2'><img src='graphics/' .
        $this->pic . "'></td></tr>");
    echo("<tr><td><font size='2'>Descrizione:</font></td><td><font
        size='2'>" . $this->description . "</font></td></tr>");
    echo("<tr><td><font size='2'>Colore:</font></td><td><font
            size='2'>" . $this->colour . "</font></td></tr>");
    }
}

class veicoloAMotore extends veicolo {
  var $fuelType;
  var $bhp;

  function veicoloAMotore($ft,$b,$d,$c,$p) {
    $this->fuelType = $ft;
    $this->bhp = $b;
    $this->veicolo($d,$c,$p);
  }

  function display2() {
    $this->display();
    echo("<tr><td><font size='2'>Tipo di carburante:</font></td>
        <td><font size='2'>" . $this->fuelType . "</font></td></tr>");
    echo("<tr><td><font size='2'>Potenza:</font></td><td><font
        size='2'>" . $this->bhp . "</font></td></tr>");
  }
}

class veicoloSenzaMotore extends veicolo {
  var $powerSource;

  function veicoloSenzaMotore($ps,$d,$c,$p) {
  $this->powerSource = $ps;
  $this->veicolo($d,$c,$p);
  }

  function display2() {
    $this->display();
    echo("<tr><td><font size='2'>Propulsione:</font></td><td><font
        size='2'>" . $this->powerSource . "</font></td></tr>");
  }
}

class bicicletta extends veicoloSenzaMotore {
  var $frameSize;
  var $gears;

  function bicicletta($ps,$fs,$g,$d,$c,$p) {
    $this->frameSize = $fs;
  $this->gears = $g;
```

```
      $this->veicoloSenzaMotore($ps,$d,$c,$p);
   }

   function display3() {
      $this->display2();
      echo("<tr><td><font size='2'>Dimensione telaio:</font></td><td><font
            size='2'>" . $this->frameSize . "</font></td></tr>");
      echo("<tr><td><font size='2'>Rapporti:</font></td><td><font
            size='2'>" . $this->gears . "</font></td></tr>");
   }
}

class motoscafo extends veicoloAMotore {
   var $engineType;

   function motoscafo($et,$ft,$b,$d,$c,$p) {
      $this->engineType = $et;
      $this->veicoloAMotore($ft,$b,$d,$c,$p);
   }

   function display3() {
      $this->display2();
      echo("<tr><td><font size='2'>Motore:</font></td><td><font size='2'>".
            $this->engineType . "</font></td></tr>");
   }
}

class automobile extends veicoloAMotore {
   var $wheelSize;

   function automobile($ws,$ft,$b,$d,$c,$p) {
      $this->wheelSize = $ws;
      $this->veicoloAMotore($ft,$b,$d,$c,$p);
   }

   function display3() {
      $this->display2();
      echo("<tr><td><font size='2'>Dimensione ruote:</font></td><td><font
            size='2'>" . $this->wheelSize . "</font></td></tr>");
   }
}

class autoSportiva extends automobile {
   var $turbo;

   function autoSportiva($t,$ws,$ft,$b,$d,$c,$p) {
      $this->turbo = $t;
      $this->automobile($ws,$ft,$b,$d,$c,$p);
   }

   function display4() {
      $this->display3();
      echo("<tr><td><font size='2'>Pressione turbo:</font></td><td><font
            size='2'>" . $this->turbo . "</font></td></tr>");
   }
}
```

```
$veh4 = new autoSportiva(2,18,"Benzina",289,"Auto
sportiva","Giallo","sportsCar.jpg");
$veh4->display4();
echo("</table>");

?>
```

Questo script utilizza un'immagine aggiuntiva per rappresentare l'auto sportiva della Tabella 38.2.

Tabella 38.2 Immagine aggiuntiva di veicolo

	sportsCar.jpg

Osservate che l'istruzione `echo("</table>");` è stata rimossa dal corppo delle classi base della struttura e spostata dopo la chiamata della funzione `display()`. L'output di questo script è quello della Figura 38.9.

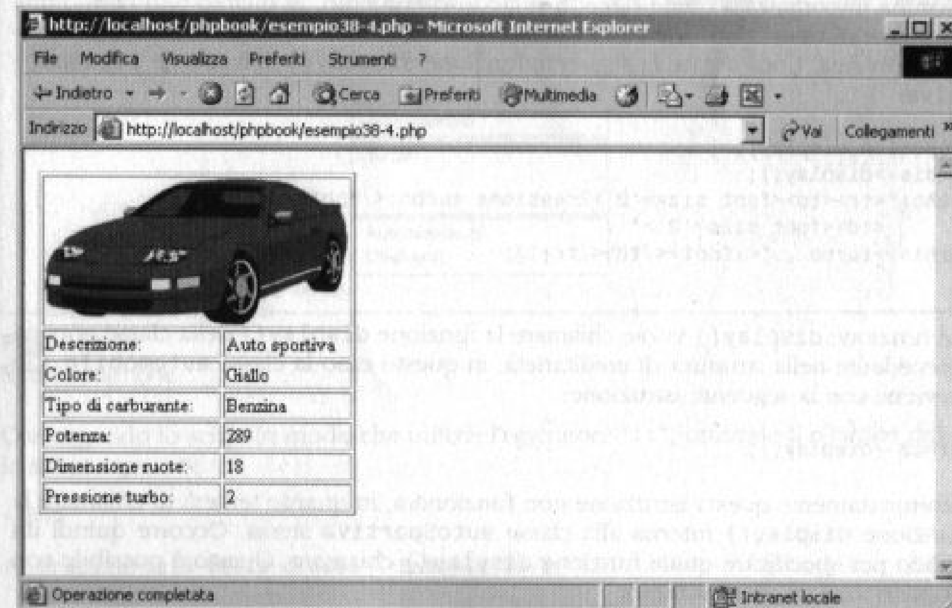


Figura 38.9
Auto sportiva.

Il problema nasce quando si vogliono creare diversi oggetti per ciascuna delle quattro classi. Benché sia possibile crearli e collocarli in un array, quando si cerca di visualizzarli con un ciclo `foreach` non si sa quale funzione `display()` chiamare, se

display3() o display4(), in quanto non si conosce il tipo degli oggetti contenuti nell'array:

```
$veh = new motoscafo("Fuoribordo","Diesel",60,"barca","Giallo",
"speedboat.jpg");
$veh2 = new bicicletta("Umana",18,21,"Bicicletta","Verde","bicycle.jpg");
$veh3 = new automobile(17,"Benzina",147,"Automobile","Rosso","car.jpg");
$veh4 = new autoSportiva(2,18,"Benzina",289,"Auto sportiva","Giallo",
"sportsCar.jpg");
$vehicles = array ($veh,$veh2,$veh3,$veh4);
foreach ($vehicles as $vehicle){
   $vehicle->display3(); // il problema si verifica qui!
   echo("</table>");
}
```

La soluzione chiama in causa il polimorfismo (che significa capacità di assumere molte forme). Dovete creare classi ereditate, ciascuna delle quali contenga funzioni con lo stesso nome. Nell'esempio, tutte le funzioni di visualizzazione devono chiamarsi display(), come mostrato nella Figura 38.10. Qui tutte le classi nella struttura di ereditarietà hanno una funzione chiamata display(). Se tutte le funzioni hanno lo stesso nome, si elimina il problema sollevato in precedenza, in quanto potete chiamare la funzione display() degli oggetti nel modo seguente:

```
$vehicle->display();
```

Non ha importanza a quale oggetto si faccia riferimento, in quanto tutti hanno una funzione di visualizzazione con lo stesso nome. Questo, tuttavia, fa sorgere un nuovo problema. Considerate la seguente funzione display() della classe autoSportiva:

```
function display() {
$this->display();
echo("<tr><td><font size='2'>Pressione turbo:</font></td>
      <td><font size='2'>" .
$this->turbo . "</font></td></tr>");
}
```

La funzione display() vuole chiamare la funzione display() della classe appena precedente nella struttura di ereditarietà, in questo caso la classe automobile. Ciò avviene con la seguente istruzione:

```
$this->display();
```

Sfortunatamente questa istruzione non funzionerà, in quanto tenterà di chiamare la funzione display() interna alla classe autoSportiva stessa. Occorre quindi un modo per specificare quale funzione display() chiamare. Questo è possibile con l'operatore "::". Per esempio:

```
automobile::display();
```

chiamerà la funzione display() della classe automobile.

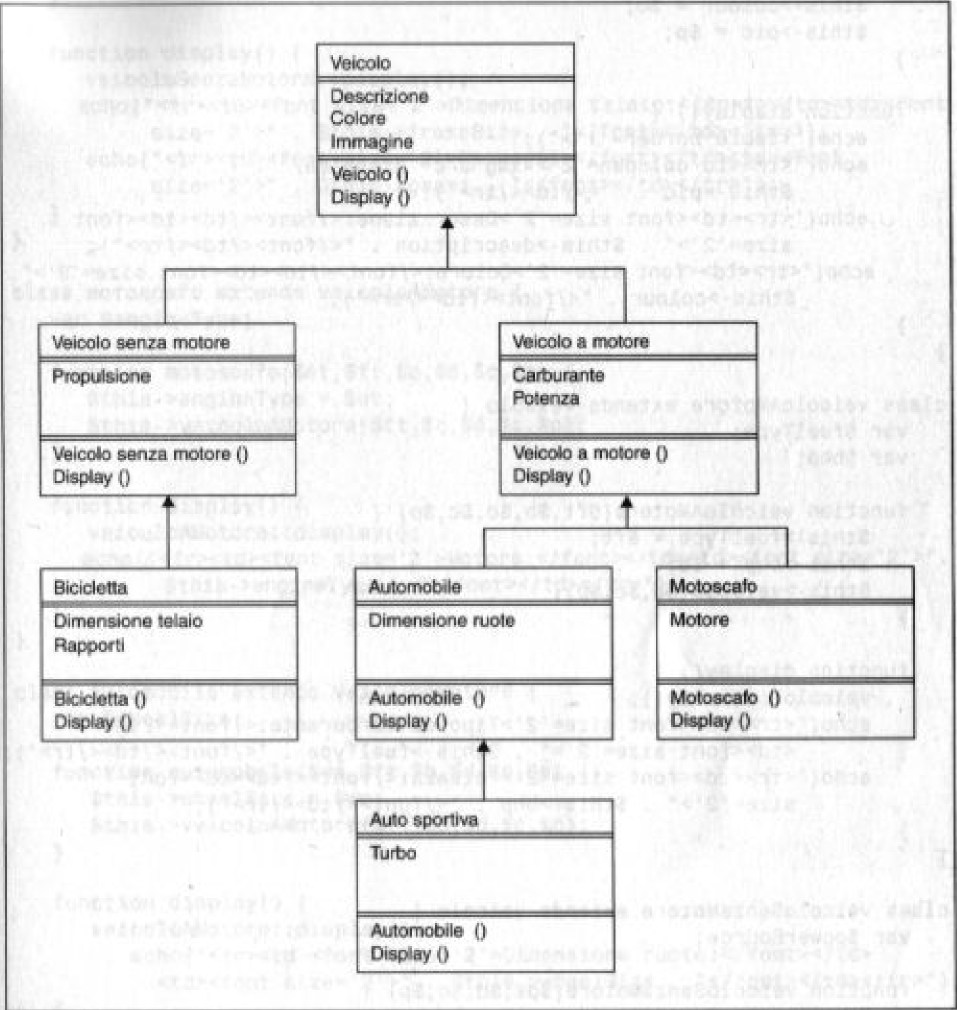


Figura 38.10
Funzioni polimorfe.

Correggendo lo script in modo che utilizzi l'operatore "::", otterrete il risultato dello script seguente.

```
<?php

// Ereditarietà delle classi - Esempio 38-5
//---------------------------------

class veicolo {
   var $description;
   var $colour;
   var $pic;

   function veicolo($d,$c,$p) {
      $this->description = $d;
```

```
      $this->colour = $c;
      $this->pic = $p;
   }

   function display() {
      echo("<table border='1'>");
      echo("<tr><td colspan='2'><img src='graphics/" .
            $this->pic . "'></td></tr>");
      echo("<tr><td><font size='2'>Descrizione:</font></td><td><font
            size='2'>" . $this->description . "</font></td></tr>");
     echo("<tr><td><font size='2'>Colore:</font></td><td><font size='2'>".
           $this->colour . "</font></td></tr>");
   }
}

class veicoloAMotore extends veicolo {
   var $fuelType;
   var $bhp;

   function veicoloAMotore($ft,$b,$d,$c,$p) {
      $this->fuelType = $ft;
      $this->bhp = $b;
      $this->veicolo($d,$c,$p);
   }

   function display() {
      veicolo::display();
      echo("<tr><td><font size='2'>Tipo di carburante:</font></td>
            <td><font size='2'>" . $this->fuelType . "</font></td></tr>");
      echo("<tr><td><font size='2'>Potenza:</font></td><td><font
            size='2'>" . $this->bhp . "</font></td></tr>");
   }
}

class veicoloSenzaMotore extends veicolo {
   var $powerSource;

   function veicoloSenzaMotore($ps,$d,$c,$p) {
      $this->powerSource = $ps;
      $this->veicolo($d,$c,$p);
   }

   function display() {
      veicolo::display();
      echo("<tr><td><font size='2'>Propulsione:</font></td><td><font
            size='2'>". $this->powerSource . "</font></td></tr>");
   }
}

class bicicletta extends veicoloSenzaMotore {
   var $frameSize;
   var $gears;

   function bicicletta($ps,$fs,$g,$d,$c,$p) {
      $this->frameSize = $fs;
      $this->gears = $g;
      $this->veicoloSenzaMotore($ps,$d,$c,$p);
   }

   function display() {
      veicoloSenzaMotore::display();
     echo("<tr><td><font size='2'>Dimensione telaio:</font></td><td><font
            size='2'>" . $this->frameSize . "</font></td></tr>");
      echo("<tr><td><font size='2'>Rapporti:</font></td><td><font
            size='2'>" . $this->gears . "</font></td></tr>");
   }
}

class motoscafo extends veicoloAMotore {
   var $engineType;

   function motoscafo($et,$ft,$b,$d,$c,$p) {
      $this->engineType = $et;
      $this->veicoloAMotore($ft,$b,$d,$c,$p);
   }

   function display() {
      veicoloAMotore::display();
     echo("<tr><td><font size='2'>Motore:</font></td><td><font size='2'>".
           $this->engineType . "</font></td></tr>");
   }
}

class automobile extends veicoloAMotore {
   var $wheelSize;

   function automobile($ws,$ft,$b,$d,$c,$p) {
      $this->wheelSize = $ws;
      $this->veicoloAMotore($ft,$b,$d,$c,$p);
   }

   function display() {
      veicoloAMotore::display();
         echo("<tr><td><font size='2'>Dimensione ruote:</font></td>
            <td><font size='2'>" . $this->wheelSize . "</font></td></tr>");
   }
}

class autoSportiva extends automobile {
   var $turbo;

   function autoSportiva($t,$ws,$ft,$b,$d,$c,$p) {
      $this->turbo = $t;
      $this->automobile($ws,$ft,$b,$d,$c,$p);
   }

   function display() {
      automobile::display();
         echo("<tr><td><font size='2'>Pressione turbo:</font></td>
               <td><font size='2'>" . $this->turbo . "</font></td></tr>");
   }
}

$veh = new motoscafo("Fuoribordo","Diesel",60,"Barca","Giallo",
"speedboat.jpg");
```

```
$veh2 = new bicicletta("Umana",18,21,"Bicicletta","Verde","bicycle.jpg");
$veh3 = new automobile(17,"Benzina",147,"Automobile","Rosso","car.jpg");
$veh4 = new autoSportiva(2,18,"Benzina",289,"Auto sportiva",
                    "Giallo","sportsCar.jpg");
$vehicles = array ($veh,$veh2,$veh3,$veh4);
foreach ($vehicles as $vehicle){
   $vehicle->display();
   echo("</table>");
}

?>
```

L'output di questo script è quello della Figura 38.11. Come potete vedere, le funzioni polimorfe consentono di visualizzare quattro diversi oggetti contenuti nell'array.

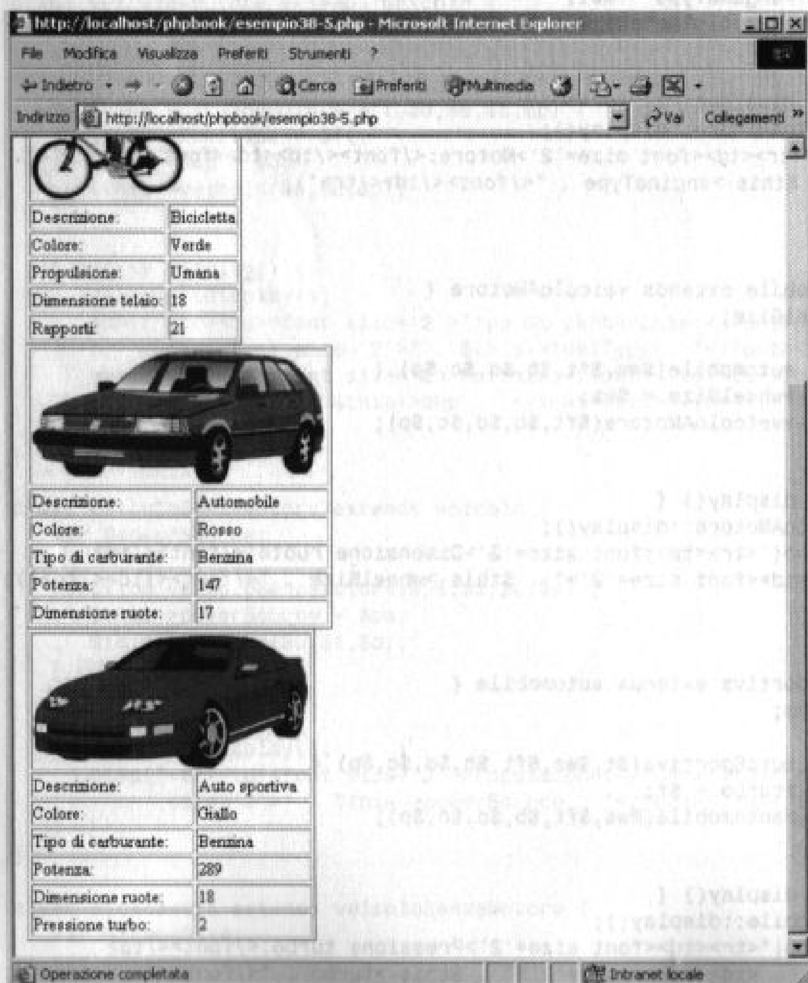


Figura 38.11
Output polimorfo.

Riepilogo

Questo capitolo ha introdotto il concetto di ereditarietà delle classi, mostrando come questa funzionalità possa essere utilizzata per creare nuove classi basate su classi già esistenti. È stato inoltre introdotto il concetto di polimorfismo, spiegando come possa aiutare a manipolare e utilizzare le classi in una struttura di ereditarietà. Nel prossimo capitolo verrà preso in esame il ruolo del commercio elettronico e si vedrà come sia possibile utilizzare PHP per creare un'applicazione Web di e-commerce.